

Application Designer

Development Workplace

Version 8.2 (2013-03-18)

March 2013

This document applies to Application Designer Version 8.2 (2013-03-18).

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2005-2013 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their licensors.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://documentation.softwareag.com/legal/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

Document ID: CIT-DEVTOOLS-82-20130318

Table of Contents

Preface	vii
I	1
1 Starting the Development Workplace	3
Starting the Servlet Container	4
Starting the Development Workplace	4
2 Elements of the Development Workplace	5
Icons at the Top	7
Open Windows	7
Navigation Frame	8
Content Frame	12
3 Project Manager	13
Invoking the Project Manager	14
Creating an Application Project	15
Removing Application Projects	15
II Layout Painter	17
4 Getting Started with the Layout Painter	19
General Information	20
Creating a Layout	21
Layout Templates	23
Opening a Layout	23
Elements of the Layout Painter	24
5 Defining the XML Layout	27
Previewing the Layout	28
Setting Properties for a Control	30
Adding Controls to the Layout	35
Changing the Appearance of Text	37
Defining a Background Image	38
Defining a Style Sheet	38
Defining Hot Keys	39
Managing the Nodes in the Layout Tree	40
Saving the Layout	43
Displaying the Layout in a New Browser Window	43
6 Using the Web Service Layout Assistant	45
Prerequisites	46
Starting the Web Service Layout	46
Loading the Web Service Description	47
Generating a Default Page Layout	48
Defining the Page Layout By Dragging the Service Elements	48
Calling the Service	50
7 Using the Code Assistant	51
Code Template	52
Opening the Code Assistant	52
Code Introspection	53

Generating the Code	54
Managing the Adapter Code	55
Saving Changes	55
Changes from the Outside	56
8 Using the Literal Assistant	57
Translation File	58
Opening the Literal Assistant	58
Maintaining Literals	59
Selecting Another Language	60
Displaying a Comparison Language	60
9 Using the XML Binding Tool	61
Opening the XML Binding Tool	62
Invoking the Property Editor	63
10 Using the Validation Rules Editor	65
Opening the Validation Rules Editor	66
Adding a New Validation Rule	67
Displaying an Overview of All Defined Validation Rules	70
11 Using the Formula Editor	71
Opening the Formula Editor	72
Adding a New Formula	73
Displaying an Overview of All Defined Formulae	75
12 Configuration, Log and Status Information	77
Preferences	78
XML Schema (XSD)	81
Protocol	81
Server Log	82
Log	84
XML	84
Resources	85
III	87
13 Layout Manager	89
Invoking the Layout Manager	90
Displaying the Layout Definitions for a Project	91
Searching for Layout Definitions	92
Selecting Layout Definitions	93
Generating HTML Pages	93
Removing HTML and XML Pages	94
Invoking the Layout Painter	95
Directories	96
Java API for HTML Page Mass Regeneration	97
14 Style Sheet Editor	99
Invoking the Style Sheet Editor	100
Defining the Location for your Own Style Sheets	101
Creating a New Style Sheet	103
Opening an Existing Style Sheet	104

Changing a Style Sheet	105
Overview of Variables	107
Regenerating Your Own Style Sheet from the Style Sheet Template	108
Using a Version Control Tool	109
15 Language Manager	111
Invoking the Language Manager	112
Defining a New Language	113
Opening an Existing Language	116
16 Literal Translator	119
Invoking the Literal Translator	120
Loading the Literals of a Layout	121
Editing the Literals	122
Adding a New Text ID	123
Removing Text IDs	123
17 WAR Packager	125
Invoking the WAR Packager	126
Types of Generation	127
Creating a Web Archive	128
Java API for WAR File Generation	129
IV Control Editor	131
18 Using the Control Editor	133
Invoking the Control Editor	134
Creating an Editor Extension	136
Adding a Control to an Editor Extension	138
Adding a Data Type to an Editor Extension	139
Deleting a Control or Data Type	141
Saving an Editor Extension	141
Opening an Editor Extension	142
Invoking Help for the Control Editor	142
19 Defining a Control	143
Attributes	144
Positioning	145
XML	147
XML Defaults	148
Protocol Extension	149
20 Examples	153
About the Examples	154
Defining a Control with a Corresponding Tag Handler	154
Defining a Macro Control	157
Additional Information	166
V	167
21 Monitoring	169
Invoking the Monitoring Tool	170
Memory	171
Active Sessions	171

- System Management 172
- 22 Layout Check 173
 - Invoking the Layout Check Tool 174
 - Displaying the Layout Definitions for a Project 175
 - Checking Selected Layout Definitions 176
- 23 Performance Tools 177
 - Recording a Performance Trace 178
 - Executing a Performance Trace 181
- 24 Developer Documents and License Information 185
 - Online Documentation 186
 - Java API 186
 - Control Reference 186
 - License 187
- 25 Special Tools 189

Preface

This documentation describes the Application Designer development tools that are available in the development workplace for creating and maintaining complex graphical user interfaces.



Note: This documentation applies to both the development workplace which is invoked in the browser and the development workplace which is available as an SWT client (SWTBasedGUI).

This documentation is organized under the following headings:

Starting the Development Workplace	Explains how to start and stop the Tomcat servlet container which comes with a Windows installation, and how to start the development workplace in the browser and in the SWT client.
Elements of the Development Workplace	Provides general information on the development workplace which is the central point for starting the development tools.
Project Manager	Describes the Project Manager which is used to manage the application projects created with Application Designer.
Layout Painter	Describes the Layout Painter which is the central tool for layout development. This also includes the tools which are embedded within the Layout Painter: Code Assistant, Literal Assistant, XML Binding tool and Validation Rules Editor and Formula Editor.
Layout Manager	Describes the Layout Manager which is used to manage the layout definitions.
Style Sheet Editor	Describes the Style Sheet Editor which is used to create style sheets.
Language Manager	Describes the Language Manager which is used to define additional languages.
Literal Translator	Describes the Literal Translator which is used to translate text IDs.
WAR Packager	Describes the WAR Packager which is used to package WAR files.
Control Editor	Describes the Control Editor which is used to build your own controls.
Monitoring	Describes the monitoring tool which is used to monitor the sessions that are currently running.
Layout Check	Describes the layout check tool which is used to validate the layout definitions.
Performance Tools	Describes the performance tools which can be used to test the performance of a web application.
Developer Documents and License Information	Describes how to invoke the online documentation, the Java API documentation and the control reference, and how to display legal notices for Application Designer.
Special Tools	Describes how to use the Unicode Transfer tool which is used to display the files which contain ASCII characters with a code higher than 127.

I

■ 1 Starting the Development Workplace	3
■ 2 Elements of the Development Workplace	5
■ 3 Project Manager	13

1 Starting the Development Workplace

- Starting the Servlet Container 4
- Starting the Development Workplace 4

The development workplace is the central point for starting tools for layout development.

Starting the Servlet Container

In order to start the development workplace, your servlet container must have been started. This can be Tomcat or any other servlet container. See also *Installation Information* in the *Installation* documentation.

Starting the Development Workplace

The development workplace can be invoked from all supported Windows and UNIX platforms.

▶ To start the development workplace in the browser

- Invoke your browser and start the development workplace with the following URL:

```
http://localhost:51000/cis/HTMLBasedGUI/workplace/ide.html
```



Note: If you have defined another port number during installation, enter this port number instead of the default port number 51000.

The development workplace is shown in your browser. See *Elements of the Development Workplace* for further information.

▶ To start the development workplace in the SWT client

- 1 Invoke your browser and enter the following URL:

```
http://localhost:51000/cis/SWTBasedGUI/index.html
```



Note: If you have defined another port number during installation, enter this port number instead of the default port number 51000.

- 2 Choose the following link: *Launch SWTBasedGUI - Development*.

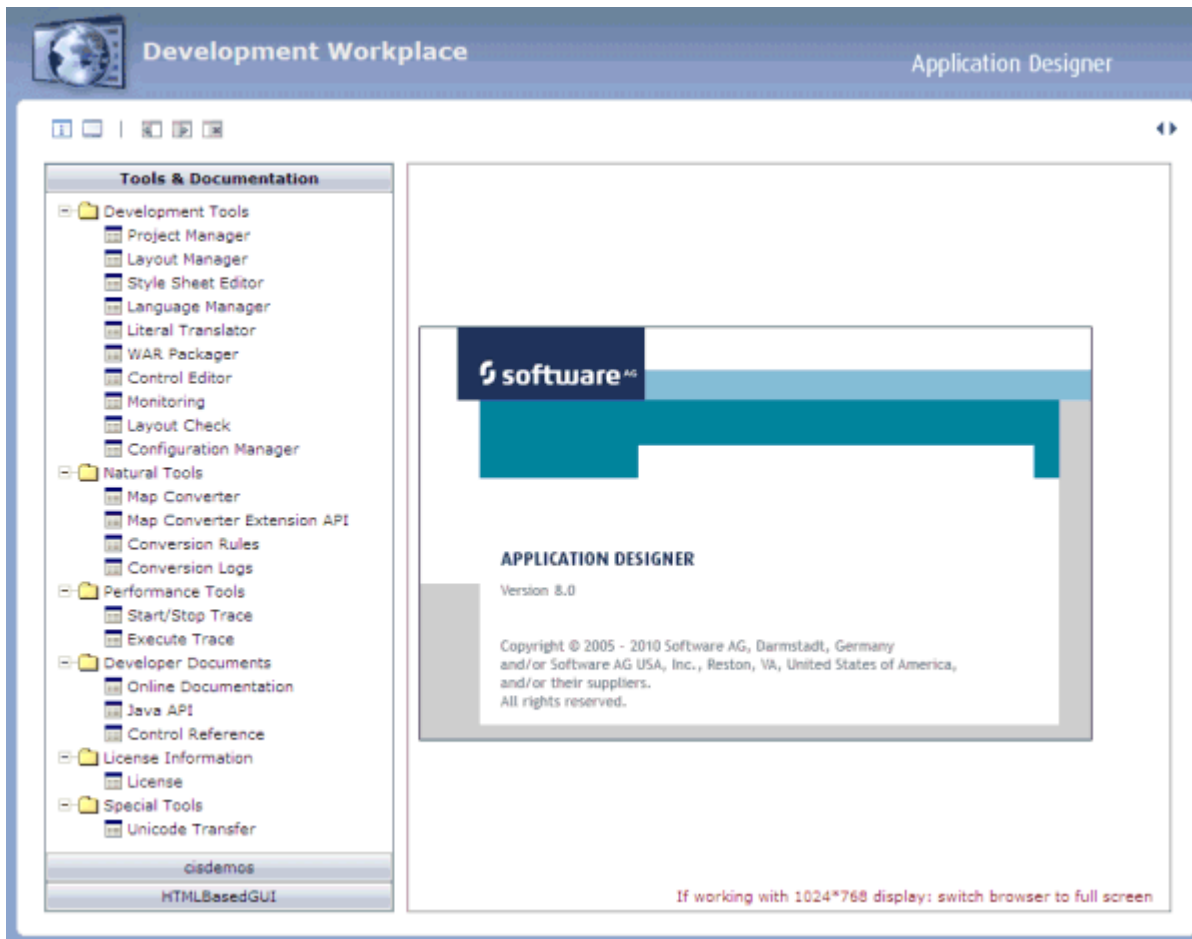
When Java Web Start has been installed, the development workplace is shown in your browser. See *Elements of the Development Workplace* for further information.

For limitations, see *SWT Client* in the *Special Development Topics* documentation.

2 Elements of the Development Workplace

▪ Icons at the Top	7
▪ Open Windows	7
▪ Navigation Frame	8
▪ Content Frame	12

When you start the development workplace, the following page appears in your browser:








Notes:

1. The **Natural Tools** node in the tree on the left is only available when Application Designer is part of a Natural for Ajax installation. These tools are not explained in this Application Designer documentation. For detailed information on how to use the Natural tools, see *Application Modernization* in the Natural for Ajax documentation.
2. The Configuration Manager is also only available when Application Designer is part of a Natural for Ajax installation. This tool, however, is explained in this Application Designer documentation. See *Customizing Configuration Files* in the *Configuration* documentation for further information.

Icons at the Top

The following icons are provided at the top of the page:

Icon	Description
	Displays information about Application Designer.
	Starts the monitoring tool .
	Closes the navigation frame on the left side of the development workplace.
	Reopens the navigation frame on the left side of the development workplace.
	Closes all open windows in the development workplace.

Open Windows

Over time, you may open a lot of windows (for example, for different layouts). A link for each open window is shown at the top of the development workplace. You can switch between these windows by choosing the corresponding link.

cisyourfirstproject/helloworld Online Documentation ◀▶

If the number of open windows exceeds the available horizontal space, you can use the following icons at the top right corner to scroll the list of windows to the right or left: ◀▶.

▶ To close a window

- Choose the standard close icon at the top right corner of the window.

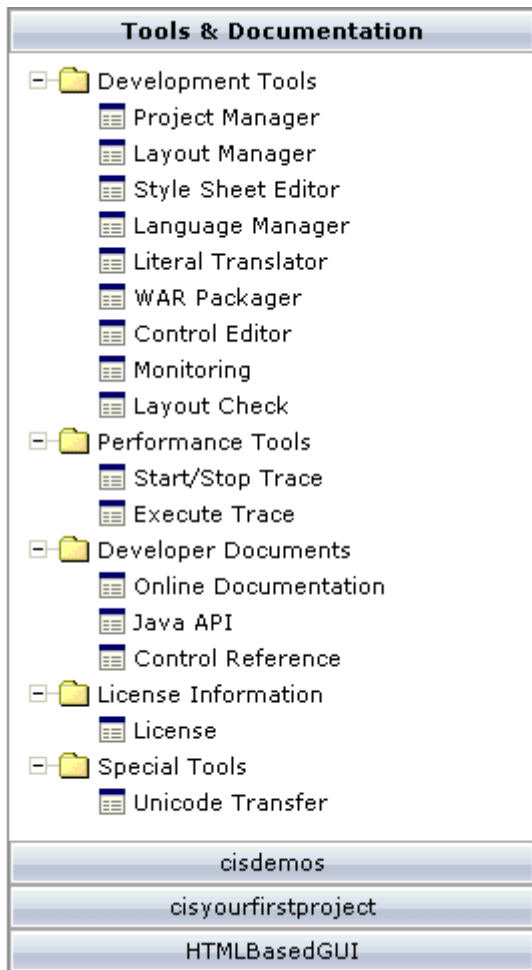
▶ To close all windows

- Choose the following icon at the top left of the development workplace.



Navigation Frame

The navigation frame contains several buttons. When you choose the **Tools & Documentation** button, a tree is shown from which you can start several tools, invoke documentation or view the license information. Each of these items is then shown in a window which appears in the area to the right of the navigation frame. These items are described in detail in the remainder of this documentation.



The other buttons in the navigation frame represent projects. The following projects are delivered with Application Designer:

- **cisdemos**

This project contains demo pages/applications that you can reach from the demo workplace.

■ HTMLBasedGUI

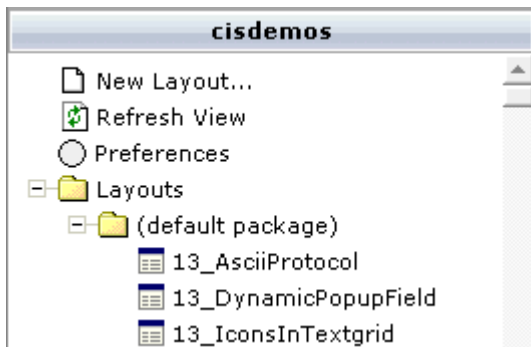
This project (together with the configuration directory *cis*) contains the Application Designer resources.



Important: Do not put your development resources into one of the projects above - create your own project(s) instead. Thus, your resource files will not be touched when you install a new Application Designer build.

Your own projects, which you create using the **Project Manager**, are also shown as buttons (for example, a project with the name "cisyourfirstproject"; see also *First Steps*).

When you choose a button which represents a project, you can manage the layouts for this project. For example:



▶ To close the navigation frame

- Choose the following icon at the top left of the development workplace.



▶ To reopen the navigation frame

- Choose the following icon at the top left of the development workplace.



The following commands are available in the navigation frame:

- [New Layout](#)
- [Refresh View](#)

- Preferences

New Layout

A layout is defined using the Layout Painter. See *Creating a Layout* for detailed information on this command.



Note: When you create a new layout, it is first shown directly below the **Layouts** node. When you refresh the view or restart Application Designer, it is automatically sorted according to your **preferences** (see below).

Refresh View

When you have added a layout directly in the file system or in a second browser window, you can refresh the view so that the latest changes to the current project are shown in the navigation frame.

▶ To refresh the view

- 1 In the navigation frame, choose the button for the project that you want to refresh.
- 2 Choose the **Refresh View** command in the navigation frame.

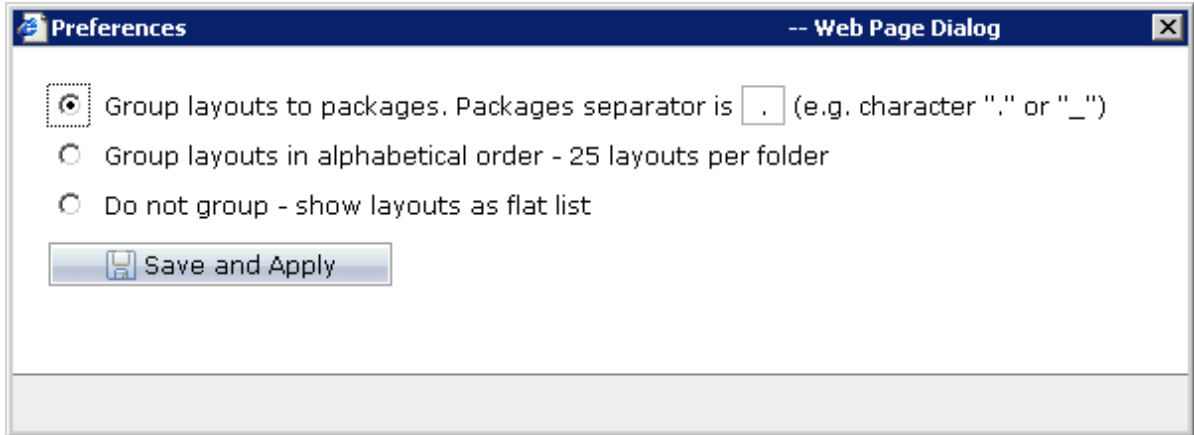
Preferences

For each project, you can define that the layouts in the **Layouts** node are grouped into folders.

▶ To set the preferences for a project

- 1 In the navigation frame, choose the button for the project that you want to refresh.
- 2 Choose the **Preferences** command in the navigation frame.

The following dialog appears.



- 3 Select one of the following options:

Group layouts to packages

The layouts are grouped into folders, according to the specified separator character.

For example, when you define an underscore (`_`) as the separator character, all layouts starting with "My_" are sorted into a folder with the name "My", and all layouts starting with "xyz_" are sorted into another folder with the name "xyz".

When the name of a layout does not contain the specified character and the layout can therefore not be sorted into a special folder, this layout is sorted into a folder with the name "(default package)".

Group layouts in alphabetical order

The layouts are sorted into folders where each folder contains a maximum of 25 layouts.

For example, you have 30 layouts and the name of the first layout in the alphabetically sorted list is "helloworld". In this case, the folder receives the name of the first layout ("helloworld") and the first 25 layouts are sorted into this folder. The next folder receives the name of your 26th layout and the remaining layouts are sorted into this folder.

Do not group

The layouts are not grouped into folders. They are shown as a flat list directly below the **Layouts** node.

- 4 Choose the **Save and Apply** button.

Content Frame

The content frame is the area to the right of the navigation frame. It displays information for the entry you have chosen in the navigation frame. For example, when you choose **Project Manager** in the navigation frame, a window pertaining to the Project Manager is shown in the content frame. Or, when you choose **Java API** in the navigation frame, a window containing the Java API documentation is shown in the content frame.

3 Project Manager

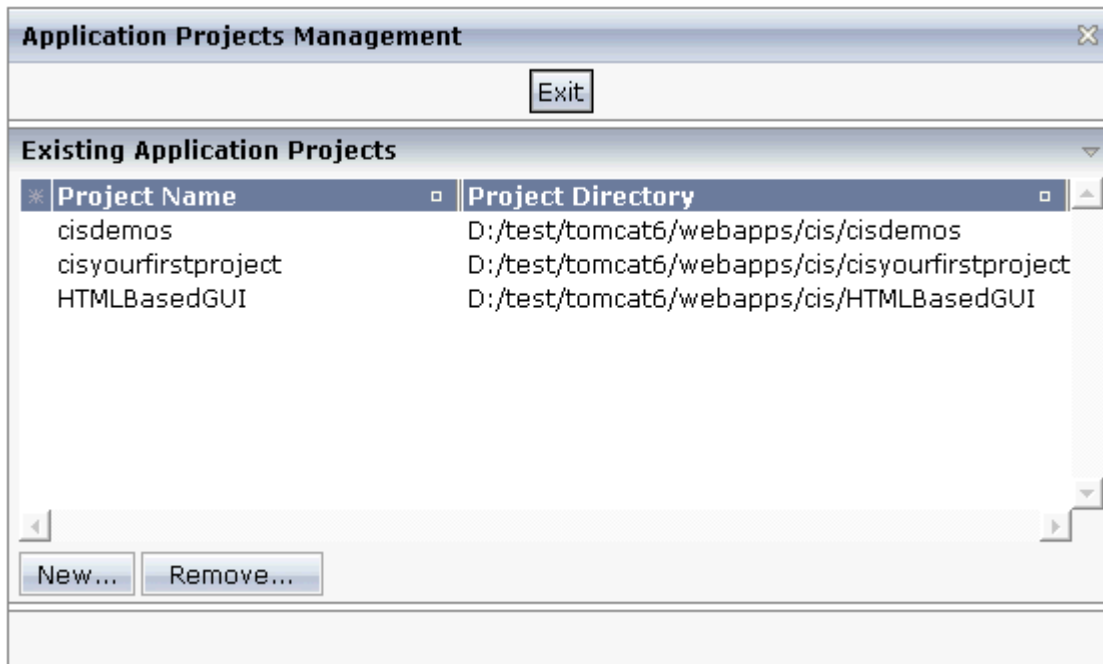
- Invoking the Project Manager 14
- Creating an Application Project 15
- Removing Application Projects 15

The resources of a web application built with Application Designer are separated into so-called “application projects”. With the Project Manager, you can create and remove application projects.

If you have larger projects and thus a high number of layouts and adapter classes, you can divide your development activities into several application projects.

Invoking the Project Manager

When you invoke the Project Manager, a list of all currently defined application projects is shown.



▶ To invoke the Project Manager

- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Project Manager**.

A list of existing application projects is now shown.

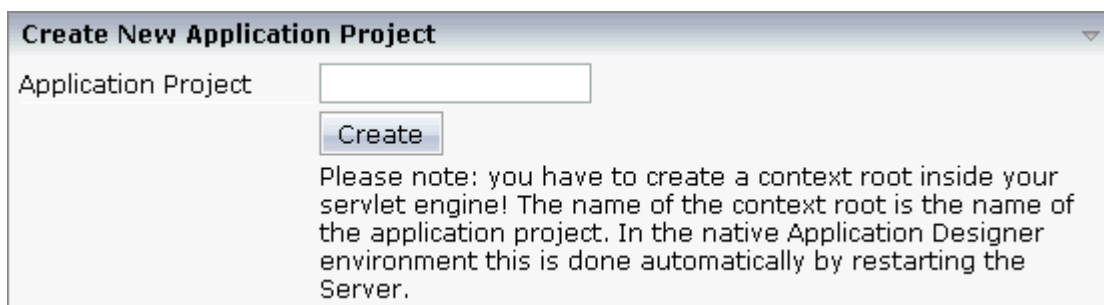
Creating an Application Project

When the list of application projects is shown, you can create new projects. A project is always created in the *<your-webapplication>* directory.

▶ To create a new application project

- 1 Choose the **New** button.

The following is shown:



- 2 Specify a project name.
- 3 Choose the **Create** button.

The directory structure is created for the new project. The new project is shown in the list of existing application projects.

The navigation frame on the left now also shows a button for your new project.

Removing Application Projects

You can remove one or more application projects at the same time.

▶ To remove one or more application projects

- 1 Select the application project in the list of application projects.

Use the **CTRL** key if you want to select more than one application project, or if you want to deselect a selected application project.

Use the **SHIFT** key to select a range of application projects.

- 2 Choose the **Remove** button.

The following is shown:



- 3 Choose the **Remove!** button.

The directory structure for this project is removed. The project is no longer shown in the list of existing application projects.

The navigation frame on the left no longer shows a button for the removed project.

II

Layout Painter

The Layout Painter is the central tool for layout development. You use it to specify the layouts of your HTML pages. The Layout Painter includes an active WYSIWYG preview in which you can test your applications. It also embeds other tools such as the Code Assistant and the Literal Assistant.

The description of the Layout Painter is organized under the following headings:

[Getting Started with the Layout Painter](#)

[Defining the XML Layout](#)

[Using the Web Service Layout Assistant](#)

[Using the Code Assistant](#)


[Using the Literal Assistant](#)

[Using the XML Binding Tool](#)

[Using the Validation Rules Editor](#)

[Using the Formula Editor](#)

[Configuration, Log and Status Information](#)

 **Note:** If you are new to Application Designer, it is recommended that you read the *First Steps* tutorial before reading the information in this part. The tutorial provides step-by-step instructions on how to create a layout with the Layout Painter.

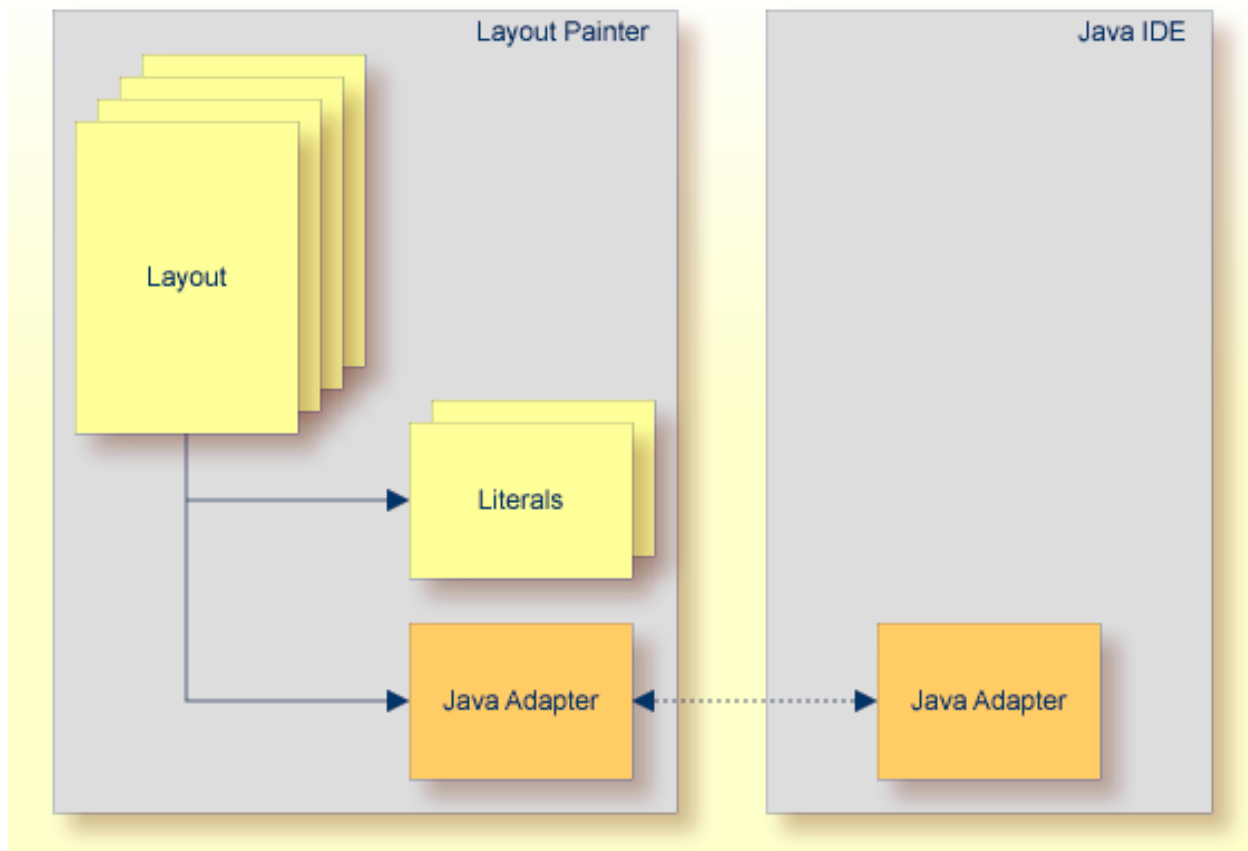
See also *Using Layout Painter Extensions* in the *Special Development Topics*.

4 Getting Started with the Layout Painter

- General Information 20
- Creating a Layout 21
- Layout Templates 23
- Opening a Layout 23
- Elements of the Layout Painter 24

General Information

Application Designer follows the commonly accepted paradigm of separating the layout (view) as much as possible from the application logic (model). You design the layout using the WYSIWYG Layout Painter. The result is kept in an XML layout description file. The HTML page is generated out of the XML layout description. On the one hand, the XML layout contains the used controls of a page; on the other hand, it keeps information about how these controls are bound to properties and methods of a server-side Java adapter. The class does not provide a large amount of logic - but connects to existing application logic. You generate the necessary property and method code inside the adapter class with the Code Assistant (which is part of the Layout Painter).



In addition, the XML layout may optionally contain so called “literals”. Literals are text identifiers (text IDs) that are replaced at runtime with language-specific texts. For each language you want to support, there must be a comma-separated value (CSV) file that contains the language-specific text per literal. You maintain these translation files using the Literal Assistant (which is also part of the Layout Painter).

Creating a Layout

You can create a layout in different ways:

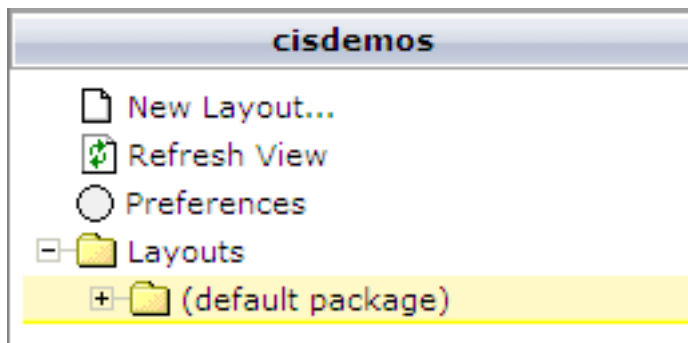
- When the Layout Painter is not yet active, you can create a layout from the navigation frame.

When you create a layout in this way, a new instance of the Layout Painter is invoked. This has the advantage that more than one layout can be active in the development workplace at the same time. When you create a layout directly from the Layout Painter, the previously active layout is closed.

- When the Layout Painter is already active, you can create a layout directly from the Layout Painter.
- You can create a layout from the Layout Manager. See *Invoking the Layout Painter* in the section *Layout Manager*.

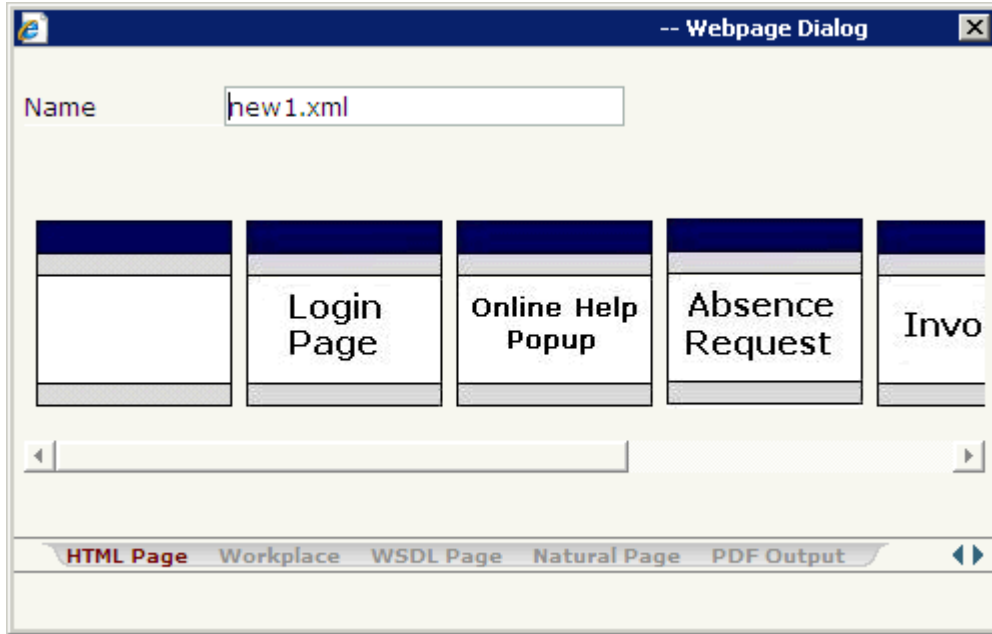
▶ To create a layout from the navigation frame

- 1 In the navigation frame, choose the button for the project that is to contain the new layout.



- 2 Choose the **New Layout** command in the navigation frame.

The following dialog appears, providing several tabs from which you can select a layout template. A default file name for your new layout is automatically provided.



- 3 In the **Name** text box, enter the name of the file that is to contain your layout definition. The name must end with ".xml".
- 4 Select the layout template that you want to use.

When you move the mouse pointer over a template, a tool tip appears indicating the type of template (for example, "HTML Page"). See also [Layout Templates](#).

The Layout Painter appears (see [below](#)).

▶ **To create a layout directly from the Layout Painter**

- 1 From the **Home** tab of the Layout Painter, choose **New Form**.

If your latest changes to your current layout have not yet been saved, a message box appears and you can decide whether to save your data, lose your data, or to go back to the unsaved layout.

When you decide to save or lose data in the above mentioned message box, or when this message box does not appear since all data have already been saved, the dialog appears in which you have to enter the name of the file that is to contain your layout definition (see above).

- 2 Enter a file name and select the template that you want to use.

Layout Templates

When you [create a layout](#), you have to select a layout template. The following types of layout templates are available:

- **HTML Page**

Different templates are available which have the PAGE control as the top node. See *Typical Page Layout* in the *Layout Elements* documentation for further information.

For information on the Online Help Pop-up template, see *Customizing the Online Help Pop-up* in *Online Help Management*.

- **Workplace**

These templates are used for arranging Application Designer pages in a frameset. They have the MFPAGE control as the top node. See *Multi Frame Pages* in the *Working with Pages* documentation for further information.

- **WSDL Page**

This template is used to build a page on base of a web service. It has the WSDLPAGE control as the top node. An assistant is provided for loading the web service description. See [Using the Web Service Layout Assistant](#) later in this documentation for further information.

- **Natural Page**

These templates are used to create pages for Natural for Ajax. They have the NATPAGE control as the top node. See the Natural for Ajax documentation for further information.

- **PDF Output**

This template is used for integrating a PDF document into an Application Designer page. It has the CISFO:FOPPAGE2 control as the top node. See the *PDF and FOP Services* documentation for further information.

Opening a Layout

You can open an existing layout in different ways:

- When the Layout Painter is not yet active, you can open a layout from the navigation frame.

When you open a layout in this way, a new instance of the Layout Painter is invoked. This has the advantage that more than one layout can be active in the development workplace at the same time. When you open a layout directly from the Layout Painter, the previously active layout is closed.

- When the Layout Painter is already active and you want to open another layout, you can open it directly from the Layout Painter.

- You can open a layout from the Layout Manager. See *Invoking the Layout Painter* in the section *Layout Manager*.

▶ **To open a layout from the navigation frame**

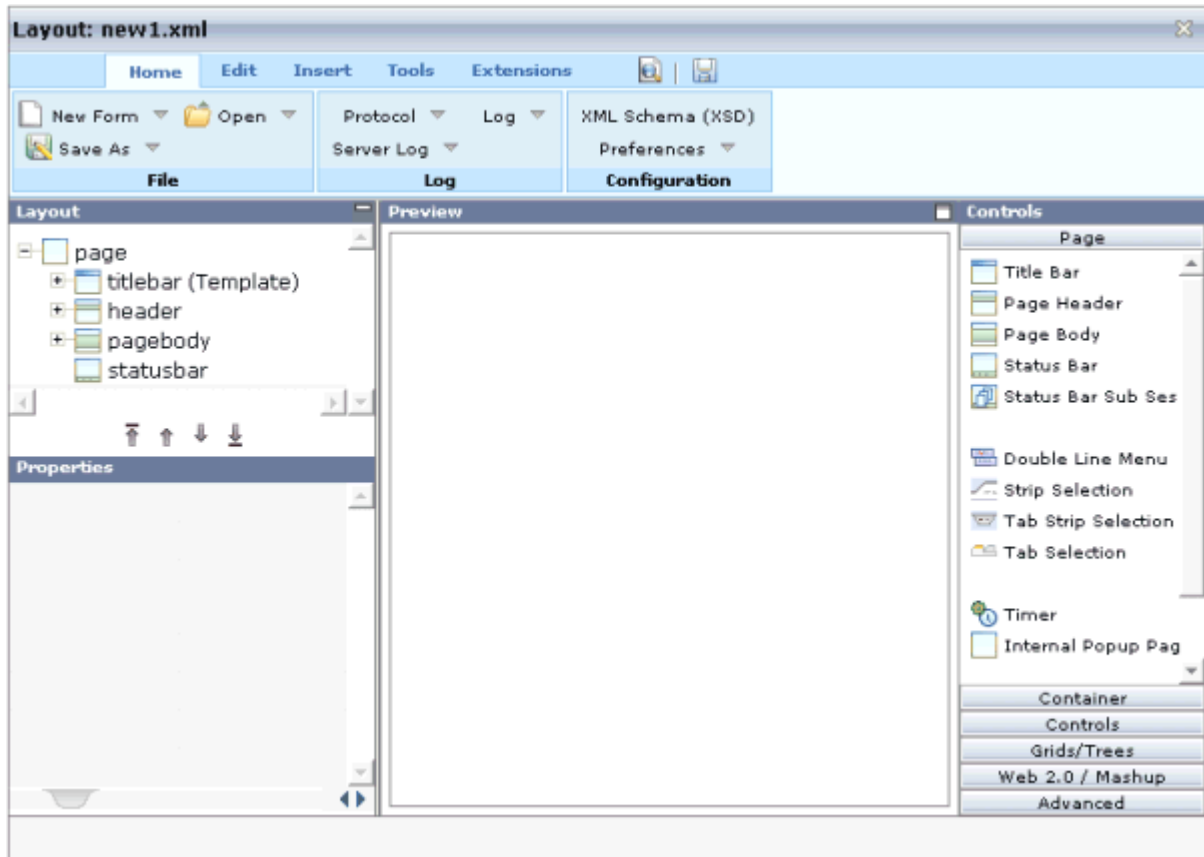
- 1 In the navigation frame, choose the button for the project which contains your layout.
The existing layouts are contained in the **Layouts** node.
- 2 If the layouts are grouped into folders, open the folder which contains your layout. See also *Preferences*.
- 3 Select the name of your layout.
The Layout Painter appears (see **below**).

▶ **To open another layout in the Layout Painter**

- 1 From the **Home** tab of the Layout Painter, choose **Open**.
If your latest changes to your current layout have not yet been saved, a message box appears and you can decide whether to save your data, lose your data, or to go back to the unsaved layout.
When you decide to save or lose data in the above mentioned message box, or when this message box does not appear since all data have already been saved, the dialog appears listing a layouts that have been defined for the current project.
- 2 Choose the layout that you want to open.

Elements of the Layout Painter

The Layout Painter appears when you **create** a new layout or when you **modify** an existing layout. The following is an example for the template type "HTML Page":



The Layout Painter is subdivided into several areas:

- [Layout Tree](#)
- [Properties Area](#)
- [Preview Area](#)
- [Controls Palette](#)

Layout Tree

The layout tree is located on the left side of the Layout Painter. It contains the controls that represent the XML layout definition. You drag these controls from the controls palette into the layout tree. Each node in the layout tree represents an XML tag.

Using the button which is located to the right of the layout tree's title bar, you can hide the layout tree and the properties area. This enlarges the preview area. To redisplay the layout tree and property area, choose the small arrow which is shown in the left of the preview area's title bar.

Properties Area

The properties area is located below the layout tree. In this area, you specify the properties for the control which is currently selected in the layout tree.

Preview Area

The preview area is located in the middle of the Layout Painter. The following can be shown in the preview area:

HTML Preview	Shows the page which is created using the controls in the layout area. This page is refreshed when you choose the preview button. See <i>Defining the XML Layout</i> for further information.
Code Assistant	Helps you to write the adapter code. You can generate <code>set/get</code> methods and event-reaction methods based on your layout definitions. See <i>Using the Code Assistant</i> for further information.
Literal Assistant	Helps you to prepare your layout for multiple languages. You can maintain texts of different languages for the literals that are referenced within the XML layout. See <i>Using the Literal Assistant</i> for further information.
XML Binding	Helps you if you want to use XML files in order to access property values. See <i>Using the XML Binding Tool</i> for further information.

Using the toggle button which is located to the right of the preview area's title bar, you can

- enlarge the preview area so that it takes up the space of the layout tree, property area and controls palette, and
- reduce the enlarged preview area so that the layout tree, property area and controls palette are shown again.

Controls Palette

The controls palette is located on the right side of the Layout Painter. It contains icons which represent the Application Designer controls. The controls palette is structured into sections (for example, **Page** and **Container**), where each section represents a certain type of control.

The sections that are shown in the controls palette and the content of a section depend on the layout template that has been used for the current layout.

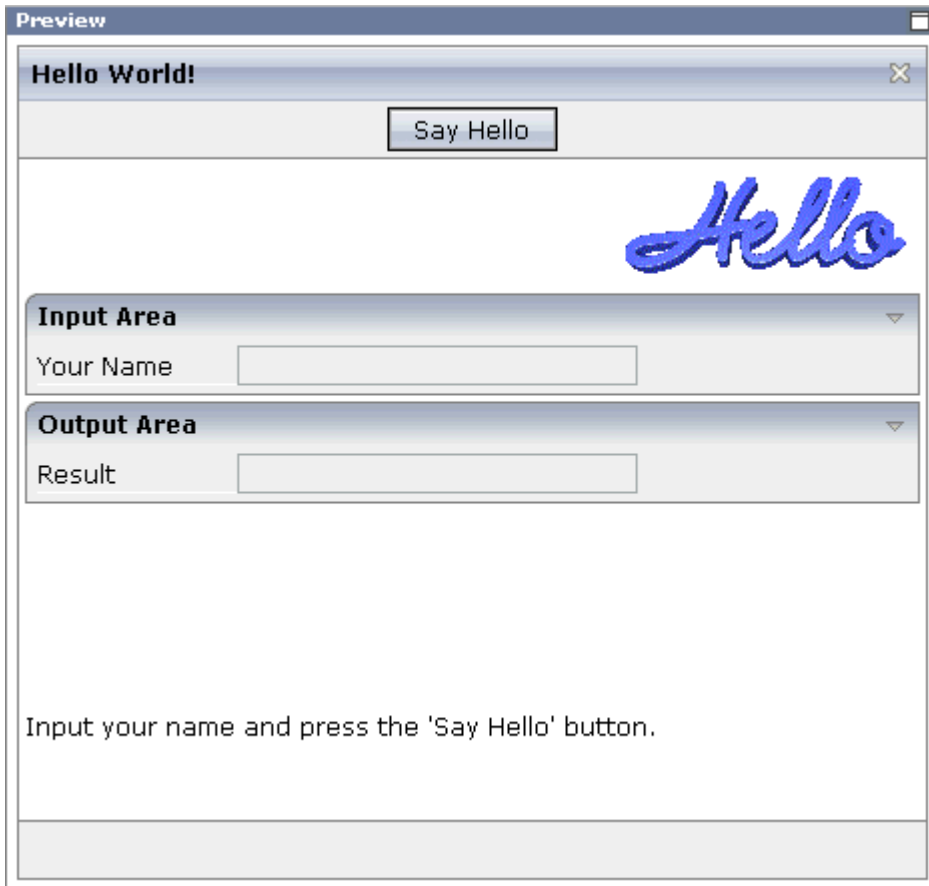
A tool tip is provided which appears when you move the mouse pointer over a control. This tool tip also displays the XML tag which will be used in the XML layout.

5 Defining the XML Layout

- Previewing the Layout 28
- Setting Properties for a Control 30
- Adding Controls to the Layout 35
- Changing the Appearance of Text 37
- Defining a Background Image 38
- Defining a Style Sheet 38
- Defining Hot Keys 39
- Managing the Nodes in the Layout Tree 40
- Saving the Layout 43
- Displaying the Layout in a New Browser Window 43

Previewing the Layout

If you want to find out how the current layout definitions are rendered on the page, you can preview the layout. The HTML preview of the layout is shown in the preview area. For example:



The preview area is a sensitive area. When you select a control in the preview area (for example, the title bar), this control is automatically selected in the layout tree.

▶ To preview the layout

- Choose the following button which is shown at the top of the Layout Painter.



The preview area is updated according to the defined preview mode (see [below](#)).

If the system finds wrong definitions, a corresponding message is displayed in the status bar. You can view the protocol to display more information about the problems that occurred when generating the preview page. See [Protocol](#) in the section *Configuration, Log and Status Information*.

When the preview button is used with one of the other tools which are shown in the preview area, the tool is refreshed so that the latest information is shown. This is helpful, for example, if the Code Assistant is currently active in the preview area and you define a new method in the properties area. If you want to display this new method immediately in the Code Assistant, you choose the preview button.

You can define that a quick preview button is to be shown in addition to the regular preview button. See [Preferences](#) in the section *Configuration, Log and Status Information*.

Preview Modes

You set the preview mode on the **Edit** tab of the Layout Painter.



Note: In the SWT client, you set the preview mode on a separate tab with the name **Preview**.

There are two different preview modes:

Run Application

This is the default setting.

As long as **Run Application** is activated, the latest changes to the adapter class (which have been made in the Java development environment) are used when you preview your layout.

Screen Test Only

When the name of an adapter class has not been defined in the `model` property of the page, or when an adapter class has been defined which does not yet exist, an error screen is shown when you try to preview your layout. In this case, you have the possibility to activate **Screen Test Only**. Your layout will then be loaded with an empty adapter. This means that you will not have any functionality. Any changes that you make in your Java development environment will not be reflected in the preview.

You can define which preview mode is to be active by default when the Layout Painter is started. See [Preferences](#) in the section *Configuration, Log and Status Information*.

Preview Configuration (SWT Client Only)

In the SWT client, the following options are available from the **Preview** tab:

SWT

When this option is selected, the preview is shown as in the SWT client.

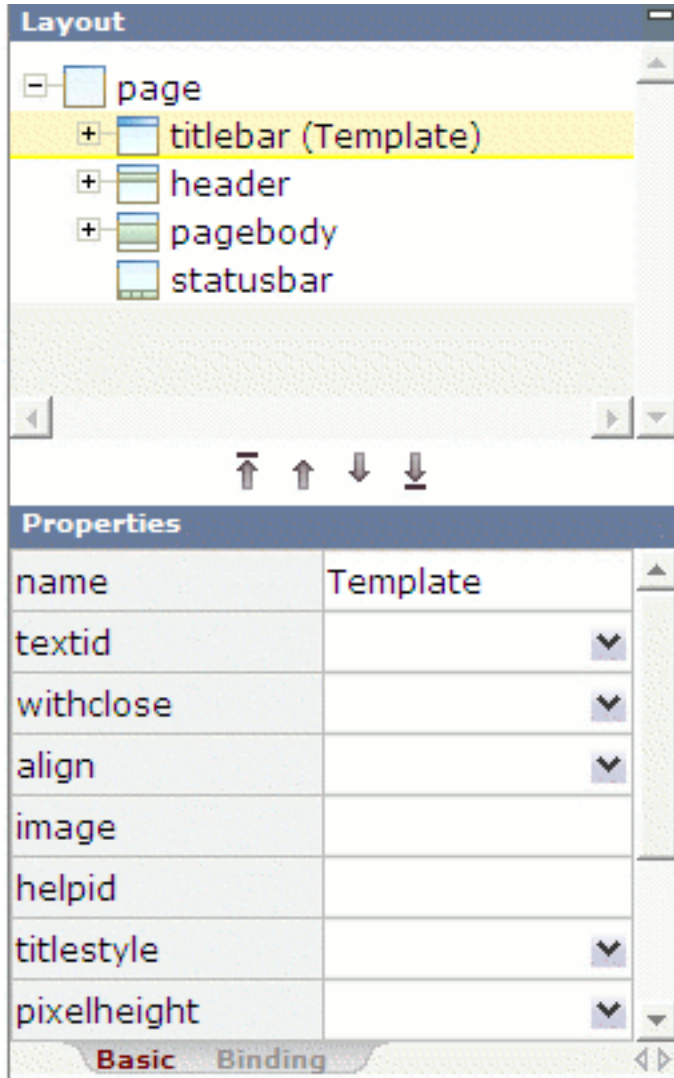
HTML

When this option is selected, the preview is shown as in the browser.


When you change the preview configuration, this is immediately considered in the preview area.

Setting Properties for a Control

When you select a control in the layout tree or in the HTML preview, its properties are shown in the properties area below the layout tree. Example:



If required, you have expand the layout tree until the control is shown for which you want to set the properties (for example, by clicking the plus sign in front of a node).

 **Note:** By clicking the icon of a node, you hide or expand the node's subnodes.

You can set properties in different ways:

- [Properties Area](#)
- [Property Editor](#)
- [Event Editor](#)

The properties are explained in the *Layout Elements* documentation in which you can view the descriptions for all properties of a control at one glance.

Properties Area

You can set the properties directly in the properties area.

▶ To set a property in the properties area

- 1 Select the node for the control in the layout tree.

Or:

Select the control in the HTML preview.

The properties for this control are now shown in the properties area at the bottom. For some properties, default entries are provided.

- 2 Specify the value for the required property.



Note: The values of some important properties (for example, the `name` or `textid` of the **Title Bar** control) are shown in the layout area: in round brackets after the XML tag. If you modify the value of such a property, the changed value will be visible when you click on the layout tree.

Property Editor

You can set the properties using the Property Editor. In this case, you can access detailed help information on each property.

▶ To set a property using the Property Editor

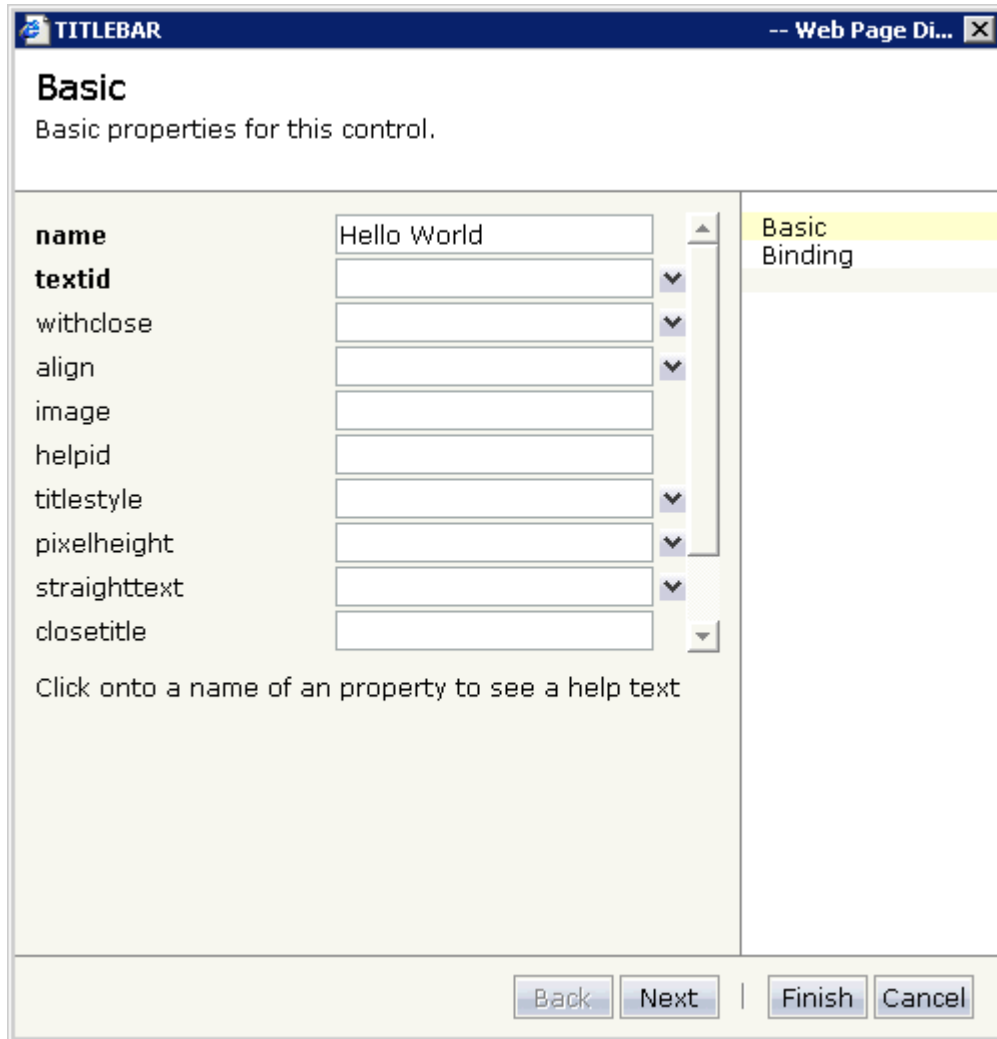
- 1 Select the node for the control in the layout tree.

Or:

Select the control in the HTML preview.

- 2 From the **Edit** tab of the Layout Painter, choose **Property Editor**.

The Property Editor dialog appears, listing all properties of the control. Example:



For some controls, the properties are spread over several pages.

- 3 If required, choose the **Next** button to display more properties for this control.

Or:

Choose the corresponding entry on the right of the dialog.

- 4 Choose the name of a property to display detailed information on this property. This information is shown below the list of properties.
- 5 Enter the required value for a property in the text box or, if available, choose the value from the drop-down list box.
- 6 Choose the **Finish** button to close the dialog.

Any changes you have applied in the dialog are saved.

Event Editor

You can set the properties using the Event Editor. The Event Editor is used in the same way as the Property Editor, but it shows only the properties which bear a reference to the program code.

▶ **To set a property using the Event Editor**

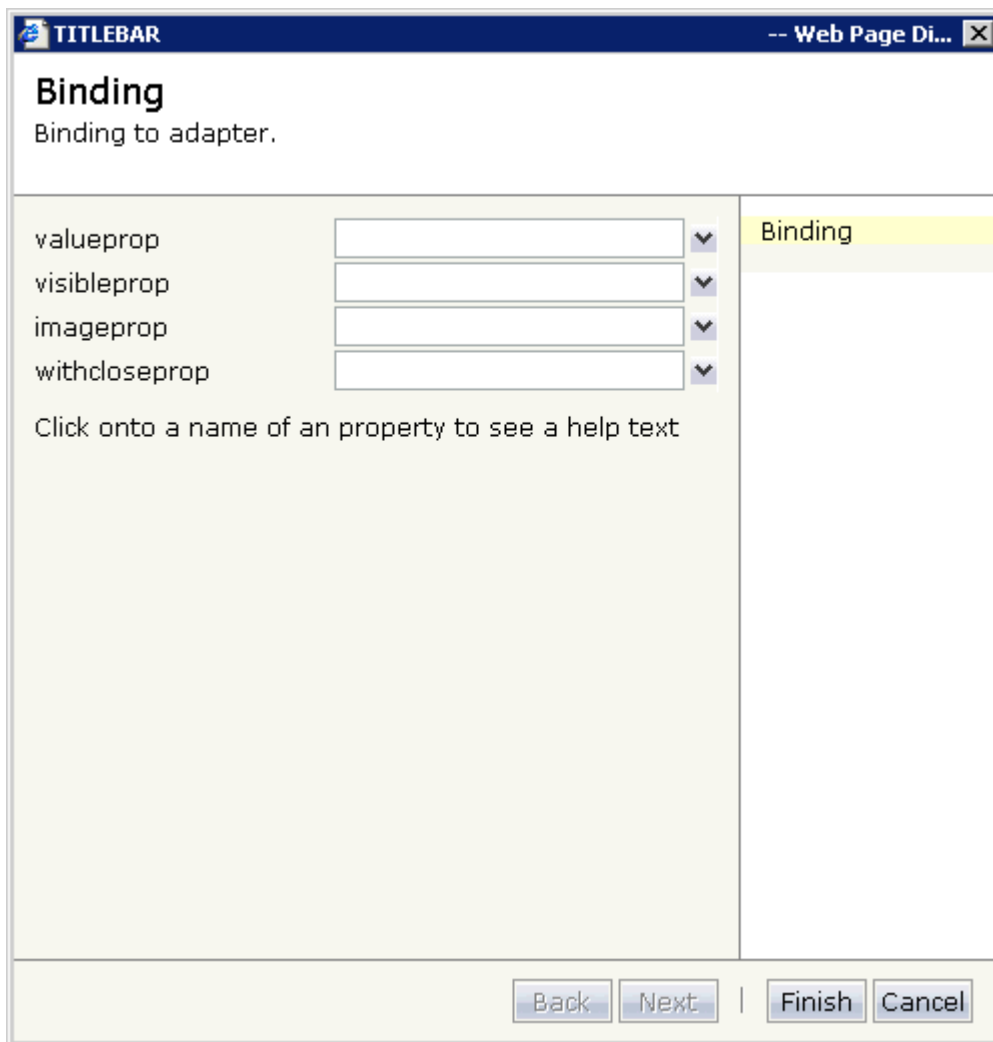
- 1 Select the node for the control in the layout tree.

Or:

Select the control in the HTML preview.

- 2 From the **Edit** tab of the Layout Painter, choose **Event Editor**.

The Event Editor dialog appears, listing only the `*prop` and `*method` properties of the control. Example:

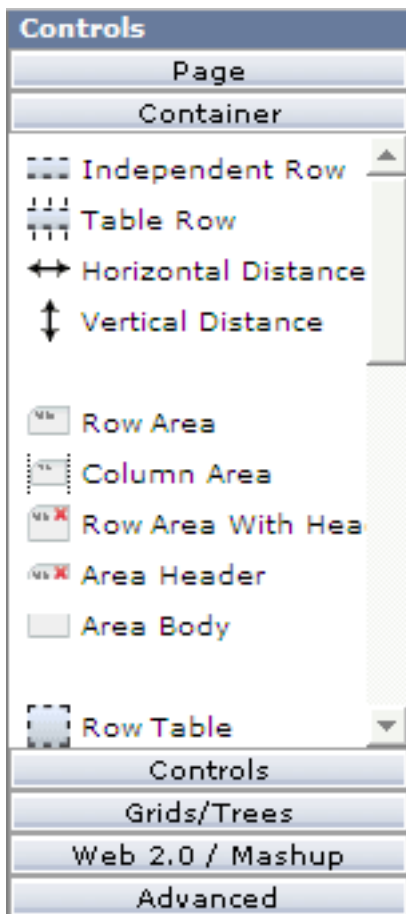


- 3 Choose the name of a property to display detailed information on this property. This information is shown below the list of properties.
- 4 Enter the required value for a property in the text box or, if available, choose the value from the drop-down list box.
- 5 Choose the **Finish** button to close the dialog.


Any changes you have applied in the dialog are saved.

Adding Controls to the Layout

The controls that you can add to the layout can be found in the various sections of the controls palette. You simply drag them from the controls palette onto the corresponding node in the layout tree or to the HTML preview. The HTML preview is immediately refreshed.



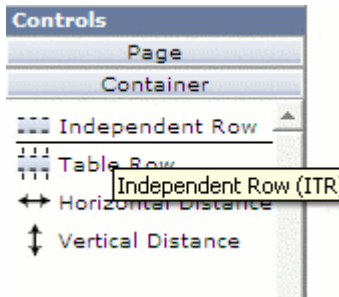
For detailed information on the controls, see the *Layout Elements* documentation.

 **Note:** You can also add controls to the layout tree as described in *Managing the Nodes in the Layout Tree*.

▶ **To add a control to the layout**

- 1 Open the required section of the controls palette by choosing the corresponding button (for example, the **Controls** button).


When you move the mouse over a control, a tool tip appears which also displays the control name which will be used in the XML layout. For example:



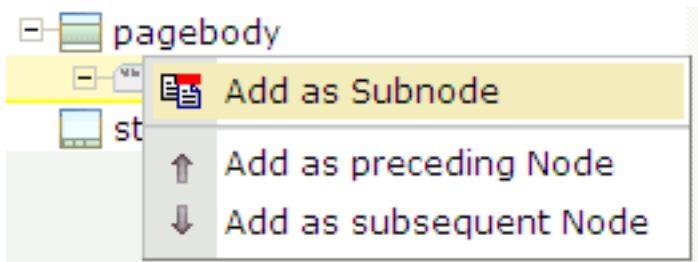
- 2 Drag the control from the controls palette onto the required node in the layout tree.


Or:

Drag the control from the controls palette and drop it in the HTML preview.

 **Note:** A control can only be inserted at a valid position. Thus, it may happen that the control is not inserted in the node which was highlighted when you dropped the control, but at a different position.

When you drop information, the system will sometimes respond by offering a context menu with certain options about where to place the control. Depending on the current context, different commands are available. For example:

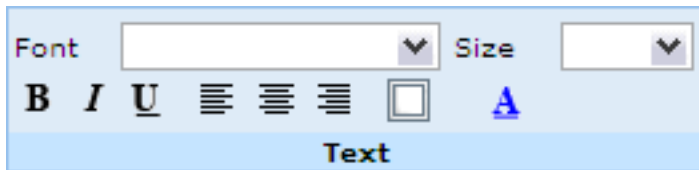


 **Note:** When you move the mouse outside the context menu, the context menu disappears. The control is not inserted in this case.

The new node is automatically selected in the layout tree so that you can maintain its properties directly in the properties area.

Changing the Appearance of Text

Instead of setting the corresponding property for a control, you can change the appearance of text (for example, on a button) using the commands in the **Text** section of the **Edit** tab.



You can change the following:

- font
- font size
- style (bold, italic, underline)
- alignment (left, center, right)
- border (not available in the SWT client)
- font color

▶ To change the appearance of text

- 1 Select the node for the control in the layout tree.

Or:

Select the control in the HTML preview.

- 2 Open the **Edit** tab of the Layout Painter and choose the desired effect in the **Text** section.

Your change is immediately shown in the HTML preview.

Defining a Background Image

Not available in the SWT client.

You can define a background image for your page. The image must be a file with the extension *gif*, *jpeg* or *jpg*.

▶ To define a background image

- 1 From the **Insert** tab of the Layout Painter, choose **Background Image**.

A dialog appears.

- 2 Specify the path to your image file or use the **Browse** button to do so.

You have to specify an image file from your local file system. The file must be located in the same directory as your web application (or any subdirectory). When you specify an absolute path, this will automatically be converted to a relative path.

- 3 Choose the **Take Over** button.

Defining a Style Sheet

Not available in the SWT client.

You can assign a style sheet to your page. This can either be a predefined style sheet or one of your own style sheets that you have created using the **Style Sheet Editor**. The style sheet must be a file with the extension *css*.

Instead of setting the corresponding property for the page (see *Static Selection of the Style Sheet File* in the *Special Development Topics*), you can define a style sheet as described below.

▶ To define a style sheet

- 1 From the **Insert** tab of the Layout Painter, choose **Style Sheet**.

A dialog appears.

- 2 Specify the path to your style sheet or use the **Browse** button to do so.

You have to specify a style sheet from your local file system. The file must be located in the same directory as your web application (or any subdirectory). When you specify an absolute path, this will automatically be converted to a relative path.

- 3 Choose the **Take Over** button.


Defining Hot Keys

You can assign hot keys to the controls PAGE, FIELD and ROWTABLEAREA2. For example, you can define the hot key CTRL+S for calling the method onSave. When hot keys can be defined for a control, a **Hot keys** tab is available in the properties area.

See also *Extended Hot Key Management* in the *Special Development Topics*.

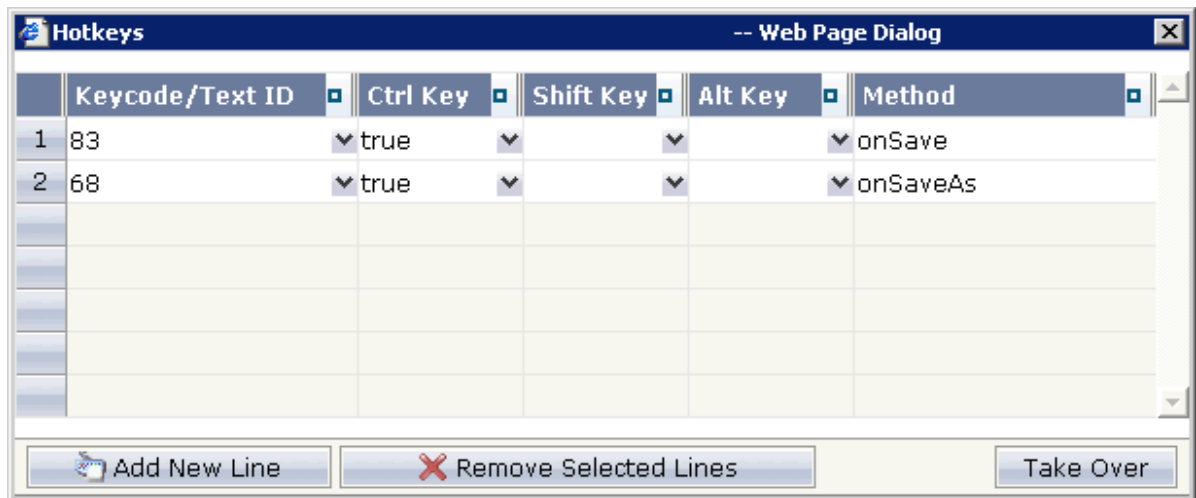
▶ To define hot keys

- 1 Select the control for which you want to define hot keys.


 **Note:** When a control is selected for which hot keys cannot be defined, your hot key definition will apply to the PAGE control.

- 2 From the **Insert** tab of the Layout Painter, choose **Hot Key**.

A dialog appears. When hot keys have already been defined, they are shown. Example:



- 3 When hotkeys have already been defined, choose the **Add New Line** button to add an additional hotkey.

 **Note:** To remove a hot key, select the corresponding row and choose the **Remove Selected Lines** button.

- 4 From the drop-down list box in the first column, select the key that you want to use as a hot key. For example, "83" represents the s key and "120" represents the F9 key.

Or:

In the text box of the drop-down list box in the first column, enter a text ID that is to be translated by the multi language management.

- 5 Define each key (CTRL, SHIFT and/or ALT) which is to be used for the hot key by selecting "true" from the corresponding drop-down list box.



Note: If neither "true" nor "false" is selected for a key, the default setting "false" is used.

- 6 In the **Method** column, specify the name of the method that is to be invoked when the user presses the hot key.
- 7 When all hot keys have been defined, choose the **Take Over** button.





Managing the Nodes in the Layout Tree

The following topics are covered below:

- [Moving Nodes](#)
- [Using the Clipboard](#)
- [Drag-and-Drop](#)
- [Context Menus](#)

Moving Nodes








You can manage the nodes in the layout tree using the following buttons which are shown below the layout tree:

Button	Description
	Moves the selected node up to first position in the tree.
	Moves the selected node up to the previous position in the tree.
	Moves the selected node down to the next position in the tree.
	Moves the selected node down to last position in the tree.

When an operation is not allowed, a corresponding message is shown in the status bar.

Using the Clipboard

You can manage the nodes in the layout tree using the following commands from the **Edit** tab of the Layout Painter:

Button	Command	Description
	Cut	Cuts the selected node.
	Copy	Copies the selected node.
	Delete	Deletes the selected node.
	Undo	Undoes the previous action. The previous action can be, for example, a move operation or a changed property value.
	Redo	Redoes the previous undo action.
	Paste as first	Pastes the cut or copied node as the first subnode of the selected node.
	Paste as last	Pastes the cut or copied node as the last subnode of the selected node.

 **Note:** To copy and paste information in the properties area, use the key combinations CTRL+C and CTRL+V.

Drag-and-Drop

You copy or move the nodes in the layout tree using drag-and-drop.

▶ To copy or move a node using drag-and-drop

- 1 Select the node you want to move so that it is highlighted.

Press **SHIFT** or **CTRL**, if you want to select more than one node.

- 2 Press the left mouse button and keep it pressed.


- 3 Drag the mouse to the target node.

While you drag the mouse, a tool tip in light colors is shown for the node(s) you are currently dragging.


- 4 Release the mouse button.

A context menu appears and you have choose one of the following commands:

Command	Description
Add a Copy in front of Node	Adds a copy of the selected node before the target node.
Add a Copy behind Node	Adds a copy of the selected node after the target node.
Move Item in front of Node	Moves the selected node and inserts it before the target node.
Move Item behind Node	Moves the selected node and inserts it after the target node.

 **Note:** When the copy or move operation is not allowed, the context menu does not appear. Instead, a corresponding message is shown in the status bar.

5 Choose the required command from the context menu.

 **Note:** To cancel the drag process, move the mouse outside the context menu. The context menu will disappear and any selected nodes will keep their current position.

Context Menus

You can manage the nodes in the layout tree using context menus. The following commands may appear in a context menu:

Command	Description
Add as first Subnode	Adds a control as the first subnode of the selected node. A cascading menu is available for this command, containing all controls that can be added as a subnode of the selected node.
Add as last Subnode	Adds a control as the last subnode of the selected node. A cascading menu is available for this command, containing all controls that can be added as a subnode of the selected node.
Add as preceding Sidenode	Adds a control before the selected node, on the same level as the selected node. A cascading menu is available for this command, containing all controls that can be added as a sidenode of the selected node.
Add as subsequent Sidenode	Adds a control after the selected node, on the same level as the selected node. A cascading menu is available for this command, containing all controls that can be added as a sidenode of the selected node.
Cut	Cuts the selected node.
Copy	Copies the selected node.
Paste as first Subnode	Pastes the cut or copied node as the first subnode of the selected node.
Paste as last Subnode	Pastes the cut or copied node as the last subnode of the selected node.
Remove	Deletes the selected node.
Properties	Invokes the Property Editor .
Open all Subnodes	Opens all subnodes of the selected node.
Close all Subnodes	Closes all subnodes of the selected node.

Saving the Layout

When you save the layout, all of your changes in the Layout Painter are saved. This includes, for example, code that has been generated with the Code Assistant and language-specific texts that have been defined with the Literal Assistant.

When you save a layout for the first time, an HTML file is generated (in addition to the XML file) which is placed into the root directory of your application project. This HTML file is updated each time you save the layout.



Note: When you save the page, the **preview area** is automatically updated.

▶ To save the page

- Choose the following button which is shown at the top of the Layout Painter.



▶ To save the page with a different name

- 1 From the **Home** tab of the Layout Painter, choose **Save As**.

A dialog box appears.

- 2 Enter the new file name. The name must end with ".xml"
- 3 Choose the **Save** button.

A copy of the previous layout is now available in the current project. It can be edited immediately.

Displaying the Layout in a New Browser Window

Not available in the SWT client.

You can display and test the current layout, which is normally shown in the preview area, in a separate browser window. The browser shows the URL that is used to invoke this page, for example, *<http://localhost:51000/cis/servlet/StartCISPage?PAGEURL=/cisyourfirstproject/helloworld.html&POPUP-TITLE=cisyourfirstproject/helloworld>*.

▶ **To display the layout in a new browser window**

- From the **Tools** tab of the Layout Painter, choose **Start in New Browser Window**.

6 Using the Web Service Layout Assistant

- Prerequisites 46
- Starting the Web Service Layout 46
- Loading the Web Service Description 47
- Generating a Default Page Layout 48
- Defining the Page Layout By Dragging the Service Elements 48
- Calling the Service 50

With the web service layout assistant, which is an editor extension, you can easily build an Application Designer page on base of a web service. You just have to provide the WSDL URL with the service description (or the WSDL file) and add input and output parameters by dragging and dropping them to your page. This is all - there is no need to provide any Java coding. Application Designer provides for the adapter to call the service.



Note: WSDL (Web Services Description Language) is an XML-based specification schema for describing the operational information of a web service. See <http://www.w3.org/TR/wsdl>.

This chapter describes all steps that are required for defining the page layout for a web service.

If you want to build your own editor extensions, see *Using Layout Painter Extensions* in the *Special Development Topics*.

Prerequisites

To run a web service with Application Designer, the following prerequisites must be fulfilled:

- The web service has to be WS-I-compliant. Non-WS-I-compliant web services are not supported. Have a look <http://www.ws-i.org/>.
- RPC/literal style web services are only supported if the operation has exactly one input part and exactly one output part.
- Outbound mapping to web service input parameters which are simple arrays is not supported. Note that the inbound mapping from simple arrays as output parameters works fine.
- Mapping of properties is not supported.

Starting the Web Service Layout

The web service layout assistant can be used with any type of **layout template**. However, when you use the template for the WSDL page, you have the additional advantage that you can immediately **call the service**.

▶ To start the web service layout assistant

- From the **Extensions** tab of the Layout Painter, choose **Web Service Layout Assistant**.

Loading the Web Service Description

When you start the web service layout assistant, the following area is shown in which you specify the web service description that is to be used.

The screenshot shows a dialog box titled "WSDL" with the following elements:

- URL/File**: A text input field.
- Search** and **Load**: Two buttons located below the URL/File field.
- Service**, **Port**, and **Operation**: Three text input fields stacked vertically.
- Use in Page** and **Use**: Two buttons located below the Service, Port, and Operation fields.
- A large empty rectangular area in the center of the dialog.
- Name**, **Nodetype**, and **Type**: Three drop-down list boxes located at the bottom of the dialog.
- A scrollbar is visible at the bottom of the dialog.

▶ To load the web service description

- 1 In the **URL/File** text box of the web service layout assistant, enter either the WSDL URL with the service description or the full path of the WSDL file and choose the **Load** button. Values from the specified URL/file are now provided in the drop-down list boxes.

Or:

If you do not know the URL/file, choose the **Search** button. A dialog appears in which you can enter the name of a WSIL (Web Services Inspection Language) file. When you choose the **Load** button in this dialog, all found WSDL and WSIL files are listed in the dialog. WSIL files can further be expanded. You can select the required WSDL file in the dialog.

- 2 Make sure that the required values are selected in the drop-down list boxes **Service**, **Port** and **Operation**.
- 3 You can now either generate a **default page layout** or you can **drag the service elements** to the layout tree. Proceed as described in the corresponding section below.

Generating a Default Page Layout

You can generate a default page layout. When you choose the preview button, the content of the layout tree is automatically updated and the corresponding controls for the web service elements can be seen in the preview area.

▶ To generate a default page layout

- 1 Load the web service description as described above.
- 2 Choose the **Use Default Layout** button.

The default page layout is generated.

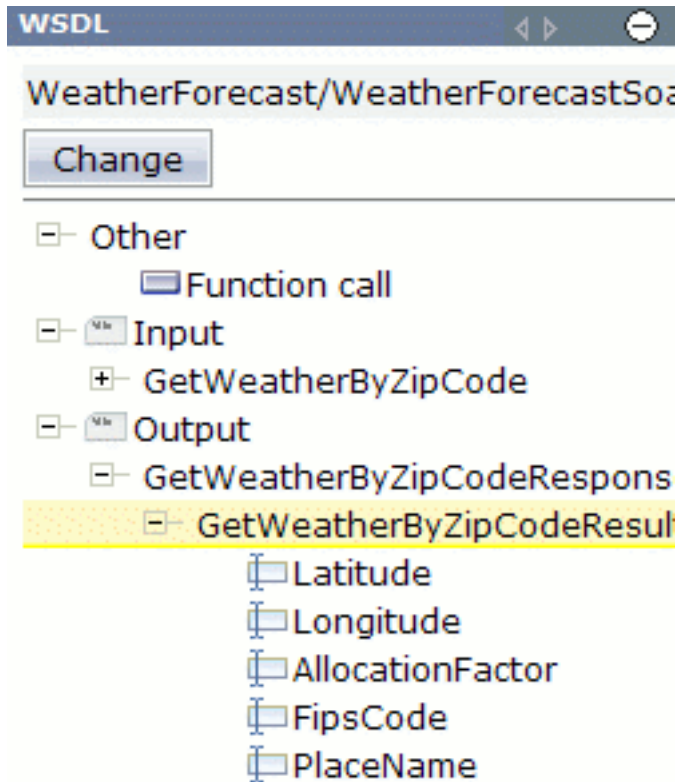
Defining the Page Layout By Dragging the Service Elements


You can drag service elements from the web service layout assistant to the layout tree.

▶ To define the page layout

- 1 Load the web service description as described above.
- 2 Choose the **Use in Page** button.





The WSDL editor parses the web service description and different information is now shown in the web service layout assistant: the service elements (**Function call** element, input parameters and service response) are shown in a tree. Example:



 **Note:** You can use the **Change** button to return to the previously shown information.

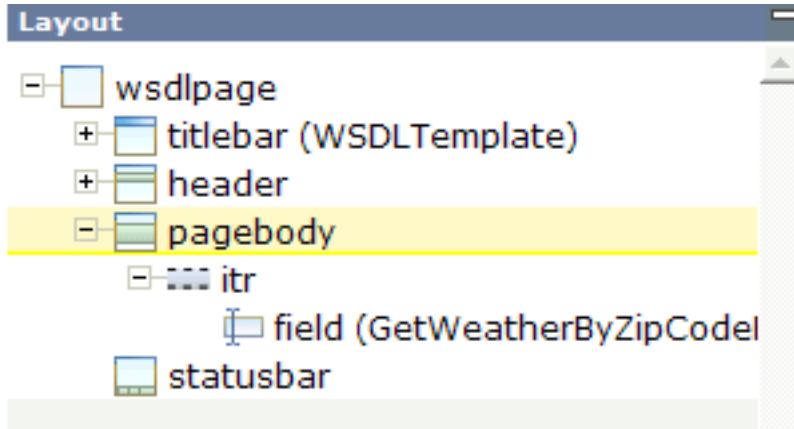
- Expand the tree that is shown in the web service layout assistant (see above) and select a service element.

Information on the selected service element is shown below the tree. The controls which can be used are listed. Example:

Name	PlaceName
Nodetype	WSDL_SIMPLE
Type	string
	COMBOFIX
	FIELD
	RADIOBUTTON
	TEXTOUT

- Drag the selected service element to the layout tree and drop it at the position of your choice.

The service element is now shown in the layout tree. Example:



- 5 Drag other parameters, the service result and the **Function call** element to the layout tree in a similar way.

Calling the Service

This feature can only be used with the **layout template** for the WSDL page.

Just choose the button that represents the **Function call** element. You do not need to provide any Java coding. The call of the web service is done by Application Designer.

7 Using the Code Assistant

▪ Code Template	52
▪ Opening the Code Assistant	52
▪ Code Introspection	53
▪ Generating the Code	54
▪ Managing the Adapter Code	55
▪ Saving Changes	55
▪ Changes from the Outside	56

The Code Assistant is part of the Layout Painter. You can use it to speed up the development of your Java adapter classes: you can generate code for the properties and methods that are referenced within the XML layout. The compilation of the adapter class is later done within your Java development environment.

Code Template

Before you open the Code Assistant, make sure that the `model` property of the page contains the correct adapter class. The default entry for this property is "DummyAdapter".

The Code Assistant looks in the source directory for the adapter class you have set with the `model` property. If the class cannot be found, the Code Assistant loads a code template. The template is stored in the Application Designer configuration directory `<your-webapplication>/cis/config` and it has the name `adapterTemplate.java`. You can change this template according to your needs.

Opening the Code Assistant

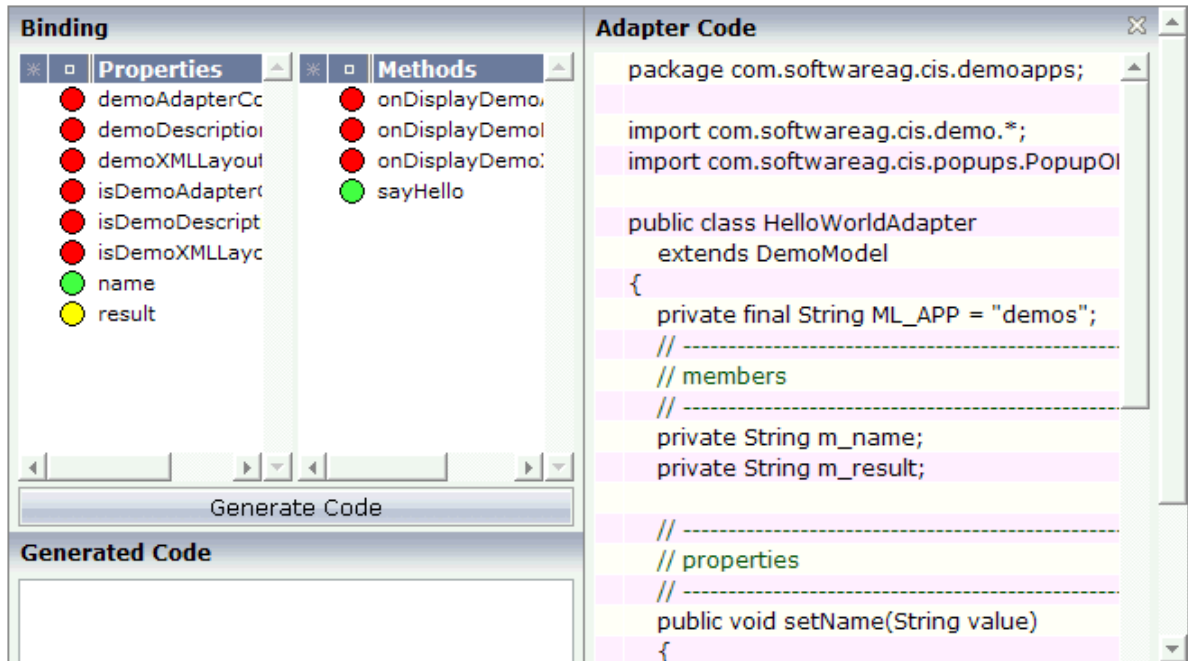
In order to read and write source code, the Code Assistant needs to know where the Java sources are kept. This information has to be specified for each project.

When you open the Code Assistant for the first time, a message appears in the status bar indicating that the Code Assistant is not available and that the Java source directory has not been set. In this case, you have to proceed as described under [Preferences](#) in the section *Configuration, Log and Status Information*.

▶ To open the Code Assistant

- From the **Tools** tab of the Layout Painter, choose **Code Assistant**.

When the Java source directory can be found, the Code Assistant is shown in the preview area. For example:



Code Introspection

The binding area at the top left of the Code Assistant lists all properties and methods that are used in the XML layout. Colored dots are shown in front of each property and method. The color of the dot indicates whether the Code Assistant could find the corresponding coding within your source code.

Introspection is done by reading and interpreting the code. Comments that highlight properties or methods are not necessary.

Possible dot colors for properties:

Green	The property's <code>get</code> and <code>set</code> methods are available.
Yellow	The <code>set</code> method is required but could not be located. The <code>get</code> method is fine.
Red	The <code>get</code> method could not be found.

Possible dot colors for methods:

Green	The method is available.
Red	The method could not be found.

When you select a property or method in the binding area, the corresponding node is automatically selected in the layout tree and the properties for this node are shown in the properties area.

In addition, when code has already been generated for the selected property or method (see below), the corresponding line is automatically selected in the adapter code which is shown at the right of the Code Assistant.

Generating the Code

The properties and methods which cannot be found in the adapter code are indicated by red dots.

When you generate code, a framework for the property or method is generated in the adapter code.

▶ To generate code

- 1 Select the property or method for which you want to generate code.

You can also use the `SHIFT` or `CTRL` key to select multiple entries.

- 2 Choose the **Generate Code** button.

The generated code appears in the area below this button.

Check whether you really want to insert these statements into your source code. You can modify or delete the code. Any code that you manually enter here will be taken over.

- 3 Optional. Choose the **+Tab** or **-Tab** button to add or remove a tab at the beginning of the code. This applies to all code which is currently shown in the **Generated Code** area.



Note: The number of spaces per tab is determined by the corresponding option in the **Configuration** dialog.

- 4 In the **Adapter Code** area on the right, select the line below which you want to insert the code.
- 5 Choose the **Take Over** button to insert the code below the selected line.

The color of the dot changes from red to green. This indicates that the corresponding methods are now available in the adapter code.

- 6 Save your changes.



Note: It is recommended that you save your changes each time you switch from the Code Assistant to another tool which may be shown in the preview area, and vice versa.

Managing the Adapter Code

You can invoke a context menu in the **Adapter Code** area. When you choose a command from this context menu, it is applied to the current selection. Using **CRTL** or **SHIFT**, you can select more than one line. Each selected line is indicated by an arrow which is shown at the beginning of the line.

The context menu provides the following commands:

Command	Description
-Tab	Removes a tab from the beginning of the selected line(s).
+Tab	Adds a tab to the beginning of the selected line(s).
Cut	Cuts the selected line(s).
Copy	Copies the selected line(s).
Paste	Only available after a cut or copy operation. Pastes the cut or copied line(s) below the currently selected line.
Remove	Removes the selected line(s).



Note: The number of spaces per tab is determined by the corresponding option in the **Configuration** dialog.

Saving Changes

You save the adapter code by choosing the **save** button. On such an event, the Code Assistant checks whether there are unsaved changes. If yes, the adapter code is written back to the file system - otherwise, the file is not touched.

Changes from the Outside

Each time you choose the **preview** button, the Code Assistant checks whether the file has changed on the file system. Therefore, the file attribute "last modified" is examined (not the file content). If the Code Assistant notices a file change, it checks whether it is holding own unsaved changes. If not, the class code is reloaded from the file system. If you have done changes in parallel (within your Java development environment and the Code Assistant) a dialog appears. Choose either to reload the file from disk (and lose the changes you have done within the Code Assistant) or to keep the Code Assistant changes (and lose the outside changes on the next save).



Note: You have to take care that the following case never occurs: the file has changed on file system and the Code Assistant has its own unsaved changes. Of course, it is reasonable to open the class both in your Java development environment and in the Code Assistant. It is recommended that you save or discard changes before you switch from the Code Assistant to your Java development environment (or vice versa). Otherwise, you will lose one of your changes.

8 Using the Literal Assistant

- Translation File 58
- Opening the Literal Assistant 58
- Maintaining Literals 59
- Selecting Another Language 60
- Displaying a Comparison Language 60

The Literal Assistant is part of the Layout Painter. You can use it to maintain texts of different languages for the literals that are referenced within the XML layout.

The concept of the multi language management is described in *Multi Language Management*. It is recommended that you read this information before you proceed with the information below.



Note: Text can also be translated using the [Literal Translator](#).

Translation File

The Literal Assistant needs to know in which file the language-specific texts are kept. Before you open the Literal Assistant, make sure that the `translationreference` property of the page contains the name for your translation file. By default, this property is empty.

For multi language support, you use the `textid` property for a label (instead of the `name` property).

See also *Writing Multi Language Layouts in Multi Language Management*.

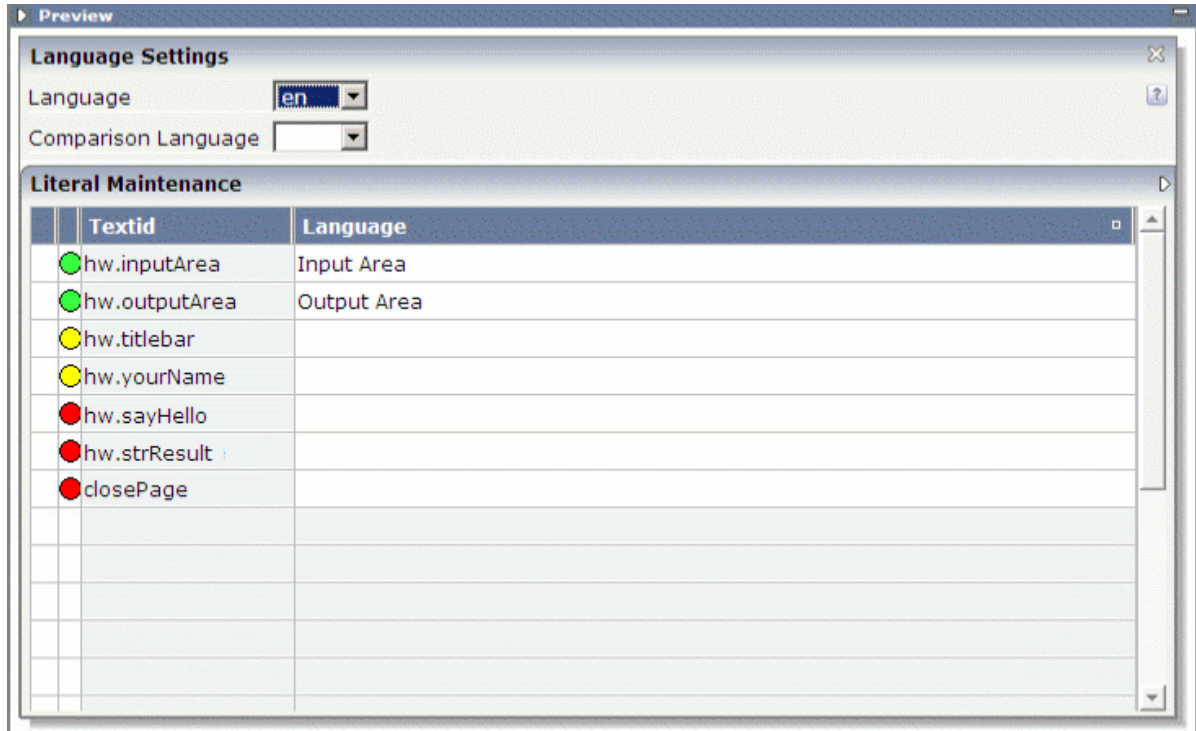
Opening the Literal Assistant

When you open the Literal Assistant for the first time, a message appears in the status bar indicating that the Literal Assistant is not available. In this case, you have to specify the name of the translation file as described above.

▶ To open the Literal Assistant

- From the **Tools** tab of the Layout Painter, choose **Literal Assistant**.

When the translation file can be found, the Literal Assistant is shown in the preview area. For example:



Maintaining Literals

All referenced literals of the currently selected language are provided in the **Literal Maintenance** area.

Colored dots are shown in front of each literal. The color of the dot indicates whether the Literal Assistant could find the corresponding text in the translation file for the currently selected language.

Possible dot colors:

Green	A text is available in the translation file.
Yellow	The translation file contains a line for the literal - but the text is empty.
Red	The literal (and text) does not yet exist in the translation file.

When you select the first column (that is, the column to the left of the colored dot), the corresponding node is automatically selected in the layout tree and the properties for this node are shown in the properties area.

▶ To add/modify text for the current language

- 1 Enter all required text in the **Language** column.

- 2 Choose the **Save** command to save your text input.

The red and yellow dots change to green, and the added/modified text is shown in the **File Content** area at the right. The translation file that is written back to the file system is sorted in ascending order.

Your changes will also be available in the preview of the HTML page.

Selecting Another Language

The valid languages are determined by the existing subdirectories in the directory `<your-project>/multilanguage`. An entry for each subdirectory is provided in the **Language** drop-down list box.



Note: You may have to choose the **Preview** command to refresh the Literal Assistant so that any subdirectories you have just defined in your file system are shown in the drop-down list box.

▶ To select another language

- From the **Language** drop-down list box, select the required language.

When you have not yet saved your previous changes, a dialog appears asking whether you want to save your changes.



Important: The Literal Assistant keeps changes for one language only. If you want to maintain texts for several languages, save the changes before you switch to another language. Otherwise, you will lose input.

The selected language is shown in the **Language** column of the table.

Displaying a Comparison Language

For comparison purposes, you can display the texts of another translation file in the **Comparison Language** column. The **Comparison Language** drop-down list box contains an entry for each subdirectory in the `<your-project>/multilanguage` directory.

▶ To display another language for comparison

- From the **Comparison Language** drop-down list box, select the required language.

The selected language is shown in the **Comparison Language** column of the table.

9 Using the XML Binding Tool

- Opening the XML Binding Tool 62
- Invoking the Property Editor 63

The XML Binding tool is part of the Layout Painter.

The property values are normally kept in the adapter class. You can use the XML Binding tool if you want to store the property values in a separate XML file. This XML file can be found in the directory `<your-webapplication>/<project>/xmldata`. It has the same name as your adapter class.

For further information, see *XML Property Binding* in the *Special Development Topics*.

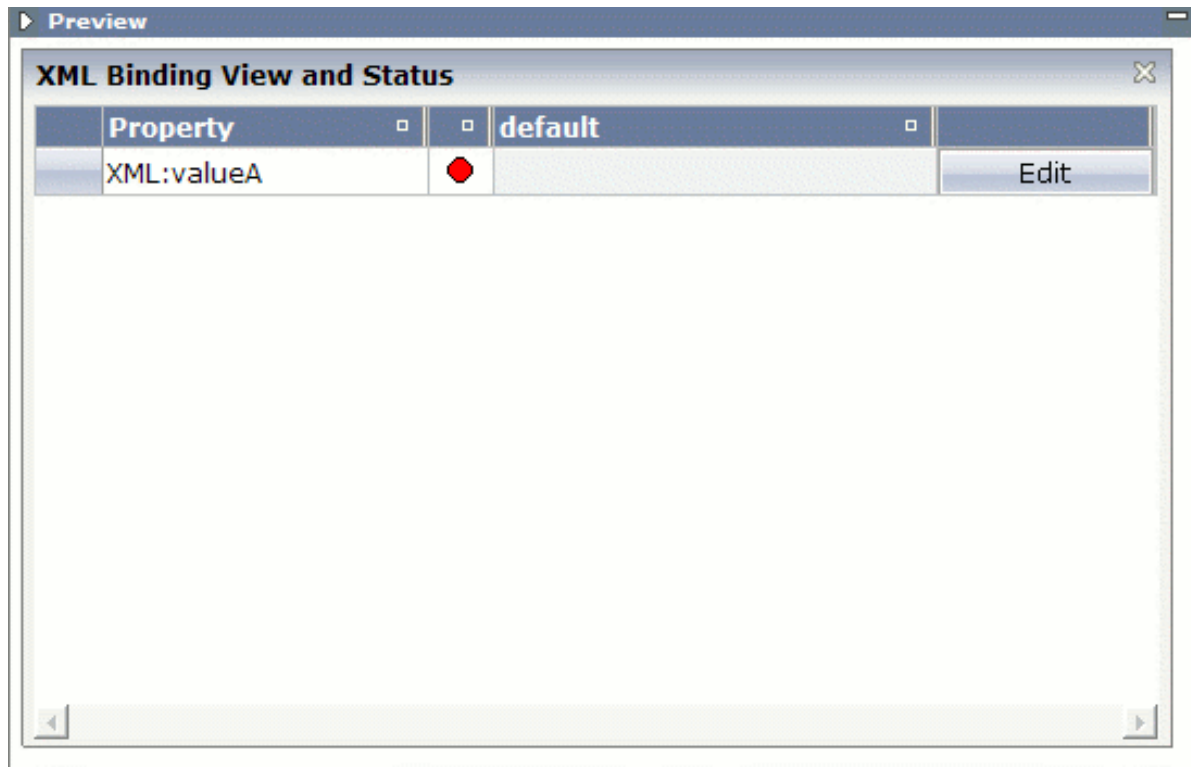
Opening the XML Binding Tool

When you open the XML Binding tool, all property values for which the prefix "XML:" has been defined in the current layout are listed in the preview area.

▶ To open the XML Binding tool

- From the **Tools** tab of the Layout Painter, choose **XML Binding**.

The XML Binding tool is shown in the preview area. For example:



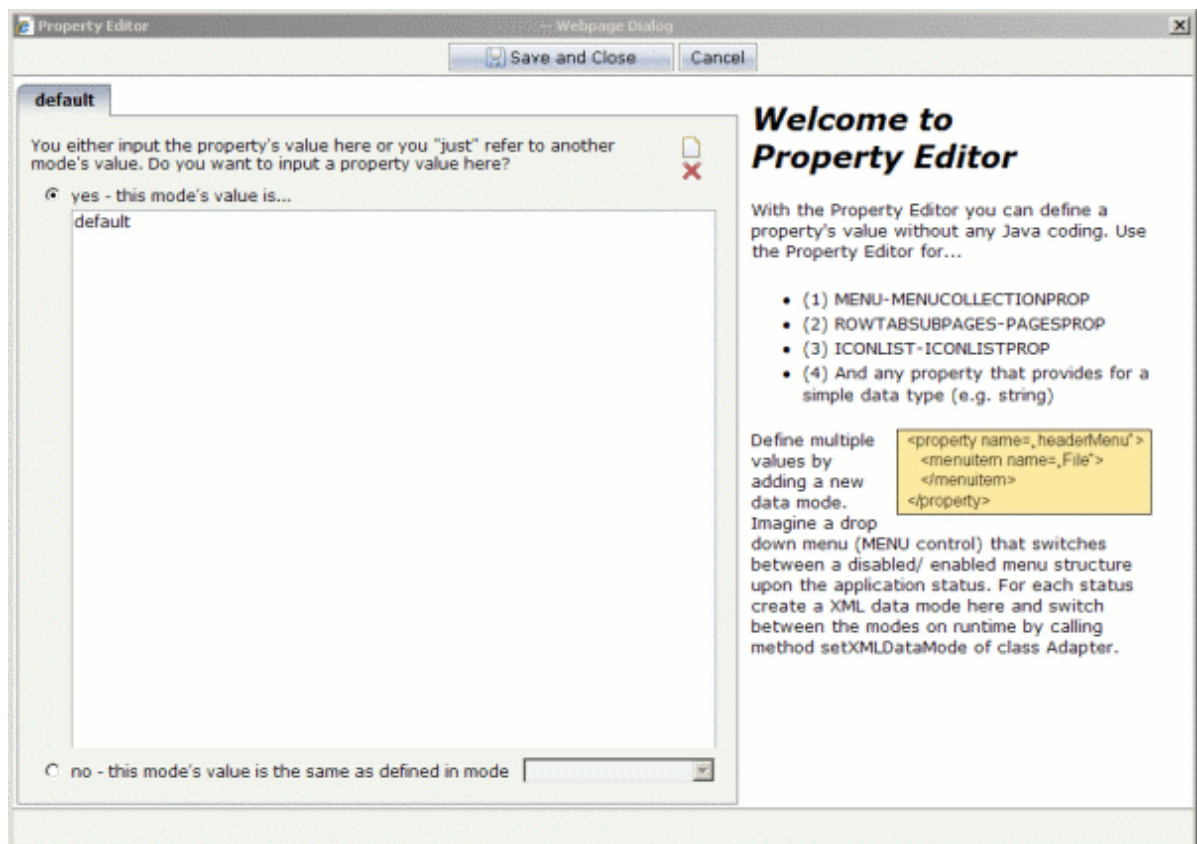
Invoking the Property Editor


You can edit the properties that are shown in the XML Binding tool.

▶ To edit a property

- 1 Choose the **Edit** button for the property that you want to change.

The following dialog appears.



 **Note:** The Property Editor can also be invoked from the properties area. It appears when you open the drop-down list box for a property value for which the prefix "XML:" has been specified.

- 2 Select one of the following option buttons:

yes - this mode's value is

When you select this option button, you have to specify the property value in the text box.

no - this mode's value is the same as defined in mode

When you select this option button, you have to specify a mode in the drop-down list box.
Or you can select an existing mode from the drop-down list box.

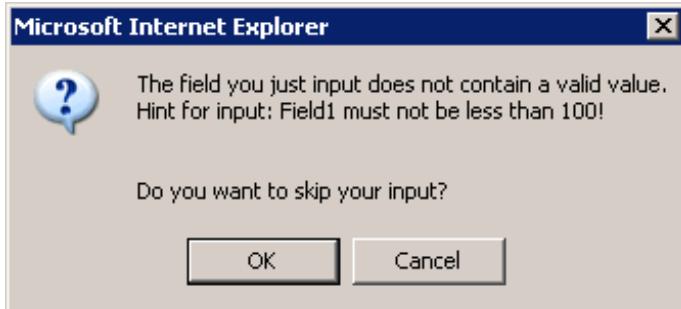
- 3 Choose the **Save and Close** button.

10

Using the Validation Rules Editor

- Opening the Validation Rules Editor 66
- Adding a New Validation Rule 67
- Displaying an Overview of All Defined Validation Rules 70

The Validation Rules Editor is part of the Layout Painter. You can use it to define that the value that is entered in a field must meet certain conditions. You can test your validation rule in the preview area. If an invalid value is entered, a corresponding message appears. Example:



 **Important:** The validation is done on the client side.

Opening the Validation Rules Editor

When you open the Validation Rules Editor, you can add, modify and remove validation rules.

▶ To open the Validation Rules Editor

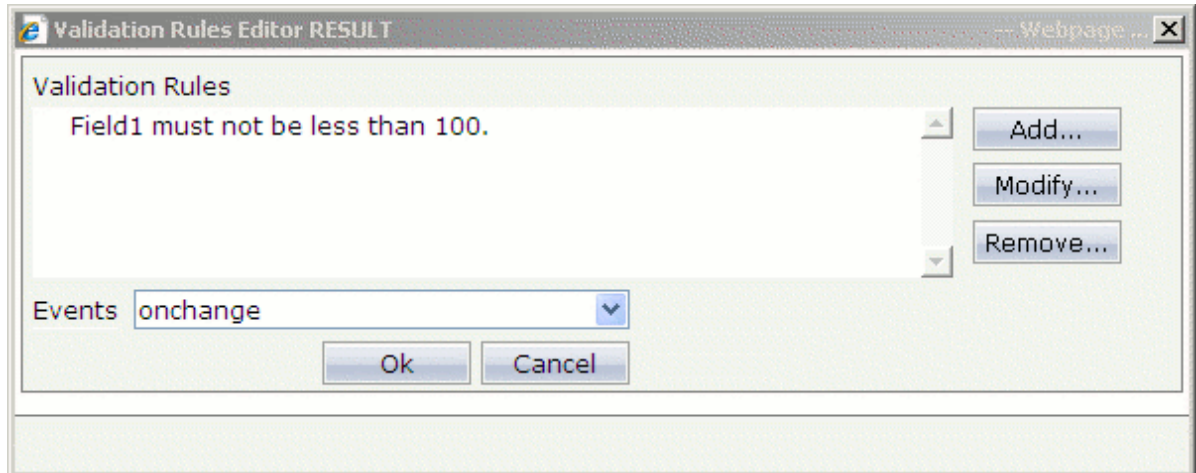
- 1 In the layout tree, select the field control that you want to validate.

Or:

Select the field control in the HTML preview.

- 2 From the **Tools** tab of the Layout Painter, choose **Validation Rules**.

A dialog appears. If validation rules have already been defined, the messages that have been defined for these rules are shown in the text box. Example:



The following command buttons are provided:

Command Button	Description
Add	Adds a new validation rule. See below.
Modify	Modifies the selected validation rule.
Remove	Removes the selected validation rule.

Adding a New Validation Rule

The prerequisite for adding a validation rule for a field is that an adapter class with the corresponding code exists for the field. Otherwise, the field does not appear in the Validation Rules Editor.

You can add more than one validation rule.

▶ To add a new validation rule

- 1 To add a validation rule, choose the **Add** button.

Additional information is now shown at the bottom of the dialog.

Validation Rules Editor RESULT

Validation Rules

Events: onchange

If this condition is true:(Enter date in YYYYMMDD format)

1				

then show this error alert:

Message:

Text ID:

- 2 From the first drop-down list box, select the name of the field for which you want to add a condition.
- 3 From the second drop-down list box, select the condition (for example, "less than").
- 4 In the third drop-down list box, enter the value for the condition (for example, "100").
- 5 Optional. From the fourth drop-down list box, select the operator **And** or **Or** if you want to define an additional condition. In this case, you have to choose the **Add** button which is shown in the lower part of the dialog and specify the required values in the resulting line (as described above).
- 6 In the **Message** text box, enter the message that is to appear in a dialog box when an invalid value is entered (for example: "The value in the field must not be less than 100!").

Or:

If you are working with several languages, enter a text ID in the **Text ID** text box. See *Multi Language Management* for further information.

- 7 From the **Events** drop-down list box, select one of the following values. This defines the behavior in the case of an error.

Event	Description
onchange	Your message will appear when the user leaves the input field.
onsubmit	Your message will appear when the information is about to be transferred to the server (for example, by choosing a Save button).

The lower part of the dialog may now look as follows:

If this condition is true:(Enter date in YYYYMMDD format)

1	name	less than	100	

then show this error alert:

Message

Text ID

Ok Cancel

- 8 Optional. If you want to delete a condition, select the corresponding row and choose the **Delete** button.
- 9 To confirm the new validation rule, choose the **OK** button in the lower part of the window.

The window is not closed and you can add further validation rules.

- 10 To confirm all validation rules and to close the window, choose the **OK** button in the upper part of the window.

Displaying an Overview of All Defined Validation Rules

You can display a list of all validation rules which are defined in the current layout.

▶ **To display an overview**


- From the **Tools** tab of the Layout Painter, choose **Overview** (located directly above the **Validation Rules** command).

A dialog appears. When validation rules have been defined in the current layout, they are listed in this dialog.

11 Using the Formula Editor

- Opening the Formula Editor 72
- Adding a New Formula 73
- Displaying an Overview of All Defined Formulae 75

The Formula Editor is part of the Layout Painter. You can use it, for example, to define that the sum of the fields A and B is to be shown in field C. In this case, the formula has to be added to field C.

 **Important:** The formula is processed on the client side.

Opening the Formula Editor

When you open the Formula Editor, you can add or remove a formula.

The prerequisite for adding a formula for a field is that an adapter class with the corresponding code exists. The Formula Editor only works with fields that have valid declared properties on the adapter side.

▶ To open the Formula Editor

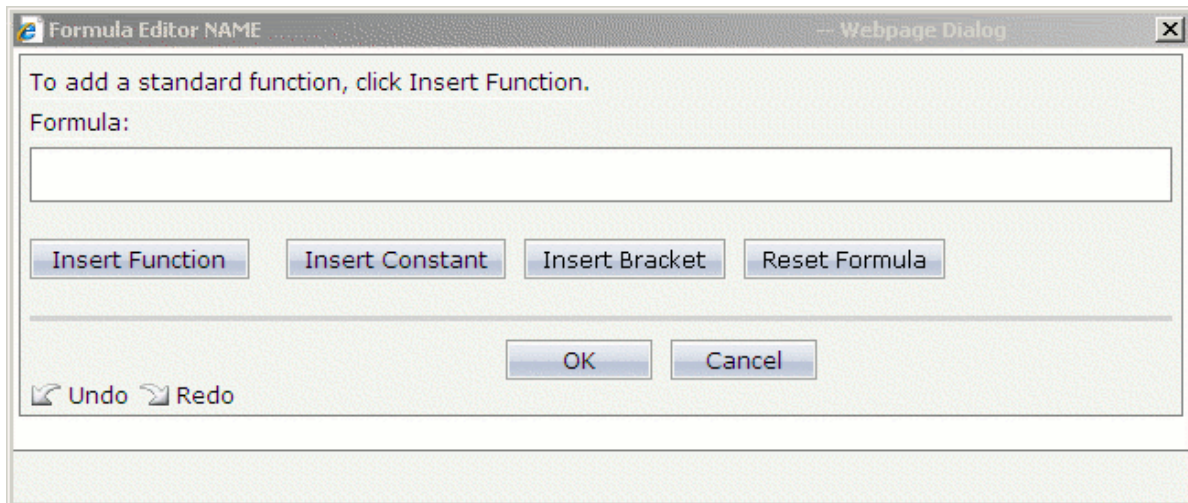
- 1 In the layout tree, select the field control for which you want to add a formula.


Or:

Select the field control in the HTML preview.

- 2 From the **Tools** tab of the Layout Painter, choose **Formula Editor**.

A dialog appears. If a formula has already been defined for the selected field, it is shown in the dialog.



 **Important:** A function is always defined from the left to the right. It is not possible to go back in the function, for example, in order to insert a missing operator. Therefore,

you should already have a plan of your function in mind before you compose the function using the controls in the Formular Editor.

The following command buttons are provided:

Command Button	Description
Insert Function	Allows you to select a function (such as sum) from a drop-down list box. After you have selected a function, you define the required fields using an "insert field" link. A simple example is provided below; see Adding a New Formula .
Insert Operator	Allows you to select an operator (such as the plus sign) from a drop-down list box.
Insert Constant	Allows you to define a constant (for example, the number "77") in a text box.
Insert Bracket	Allows you to select brackets from a drop-down list box.
Reset Formula	Deletes the formula.

Adding a New Formula

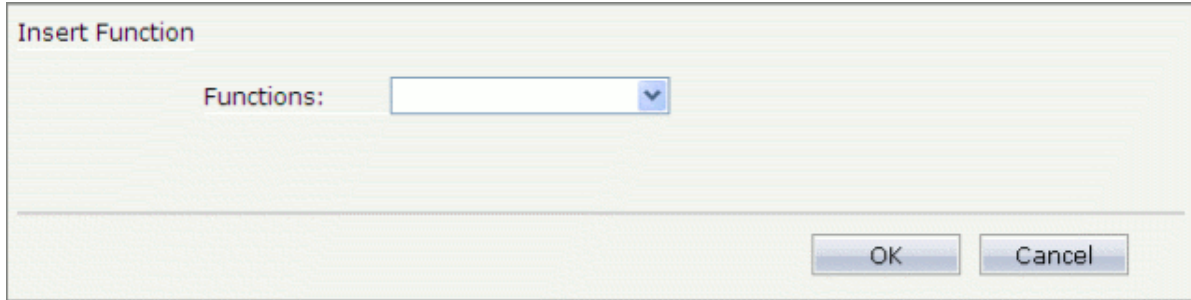
This section explains how to define a simple formula: the sum of the fields **Value A** and **Value B** is to be shown in the **Sum** field.

▶ To add a simple formula

It is assumed that the Formula Editor has been invoked as described [above](#) (the field was selected in which the sum is to be shown).


- 1 Choose the **Insert Function** button.

The following information is now shown at the bottom of the dialog.



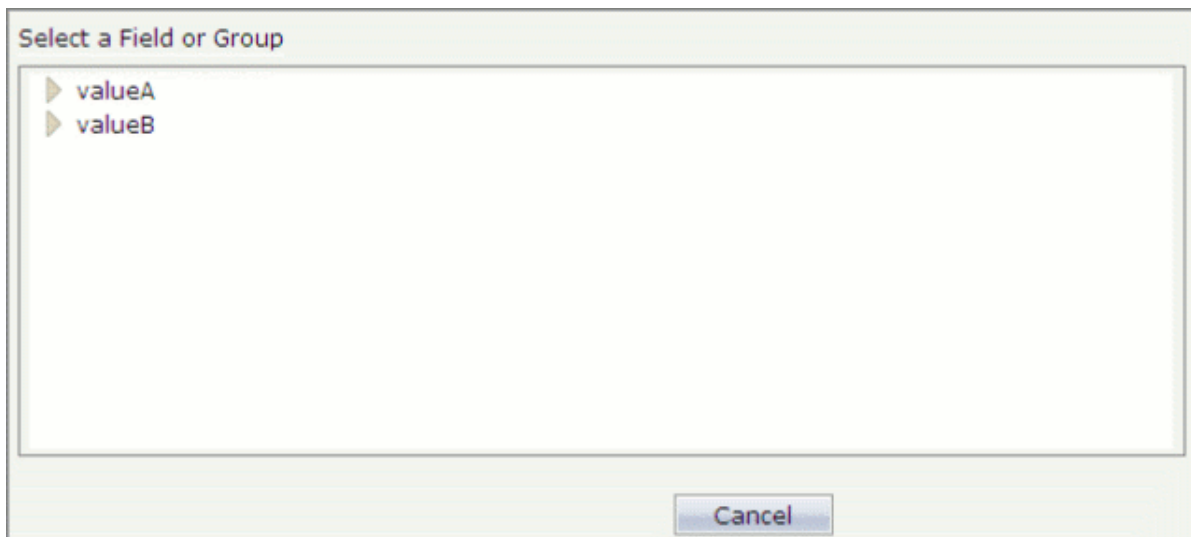
- 2 Select the required function (for this example, select **sum**) from the **Functions** drop-down list box and choose the **OK** button.

The information at the bottom of the dialog is no longer shown. The selected function is now shown in the **Formula** text box, together with the link "insert field".

 **Note:** It is not possible to edit the formula in the **Formula** text box.

- 3 Choose the "insert field" link in the **Formula** text box.

The fields that are currently defined in the layout are now shown at the bottom of the dialog. Example:



- 4 Choose a field (for this example, choose **valueA**).

The information at the bottom of the dialog is no longer shown. The selected field is now shown in the **Formula** text box. The link "insert field" is still shown in case you want to insert more fields.

- 5 Insert all required fields as described above (for this example, choose **valueB**).

For this example, the formula should look as follows:

```
sum(valueA,valueB)
```

- 6 When the formula is complete, choose the **OK** button.

The Formula Editor is closed. You can now test your formula in the preview area.

Displaying an Overview of All Defined Formulae

You can display a list of all formulae which are defined in the current layout.

▶ To display an overview

- From the **Tools** tab of the Layout Painter, choose **Overview** (located directly above the **Formula Editor** command).

A dialog appears. When formulae have been defined in the current layout, they are listed in this dialog.

12 Configuration, Log and Status Information

▪ Preferences	78
▪ XML Schema (XSD)	81
▪ Protocol	81
▪ Server Log	82
▪ Log	84
▪ XML	84
▪ Resources	85

Preferences

You have to define the source directory into which the adapter classes for the current project are to be written. This directory is used by the [Code Assistant](#).

▶ **To view/modify the configuration**

- 1 From the **Home** tab of the Layout Painter, choose **Preferences**.

A dialog appears. For example:



The name of the configuration file and the path to the current project are shown at the top of the dialog.

The following options are available:

Source Directory

The path to your Java sources. This text box is initially empty.

It is not mandatory that your sources are stored in the *<your-webapplication>* directory. They can also be stored in another directory.

Spaces Per Tab

The number of spaces that are to be inserted into the code when you add tabs with the Code Assistant.

Property Method Within One Line

When this check box is selected, source code is written as follows:

```
// property >fieldText5<
String m_fieldText5;
public String getFieldText5()
{
    return m_fieldText5;
}
public void setFieldText5(String value)
{
    m_fieldText5 = value;
}
```

When this check box is not selected, source code is written as follows:

```
// property >fieldText5<
String m_fieldText5;
public String getFieldText5() { return m_fieldText5; }
public void setFieldText5(String value) { m_fieldText5 = value; }
```

Use Prefix

The prefix that is to be used for all properties in the adapter class. By default, the prefix "m_" is defined. Example: "m_fieldText5".

Member Qualifier

Using the options in this drop-down list box, you can specify whether the package is private, protected or public. Example:

```
protected String m_fieldText5;
```

When none of these options is explicitly defined in this text box, the package is public by default. Example:

```
public String m_fieldText5;
```

Disable Validation

When this check box is selected, properties and methods are not validated.

When this check box is not selected (default), properties and methods are validated. In this case, the regular expressions listed in the **Regular Expression** text box (read only) are used for the validation.

Java Classname

You can specify a Java class which implements the interface `IEditorValidValuesProvider`. For details on this interface, see the Java API documentation. You have to define the package, without the extension ".java". For example:

```
com.softwareag.cis.Testclass
```

Quick Preview

You can define whether the quick preview button is to be shown at the top of the Layout Painter, in addition to the regular preview button. The following check boxes are available:

Show	The quick preview button is shown. When you choose this button, the changes to the adapter class are not applied. Only the layout in the preview area is updated.
Hide	The quick preview button is not shown (default).

See also [Previewing the Layout](#).

Preview Mode

You can define which mode is to be active by default when the Layout Painter is started. The following check boxes are available:

Screen Test Only	Loads the layout with an empty adapter.
Run Application	The latest changes to the adapter class are used (default).

See also [Previewing the Layout](#).

- 2 Choose the **Save and Apply** button to save your changes.

XML Schema (XSD)

You can create a ZIP file which includes an up-to-date XML (XSD) schema.

If required, you can save and extract the ZIP file and load the included *editor.xsd* file into an editor such as XML Spy in order to validate an XML layout which has been coded manually.

▶ To create a ZIP file

- From the **Home** tab of the Layout Painter, choose **XML Schema (XSD)**.

A dialog appears asking whether you want to open or save the ZIP file.

Protocol

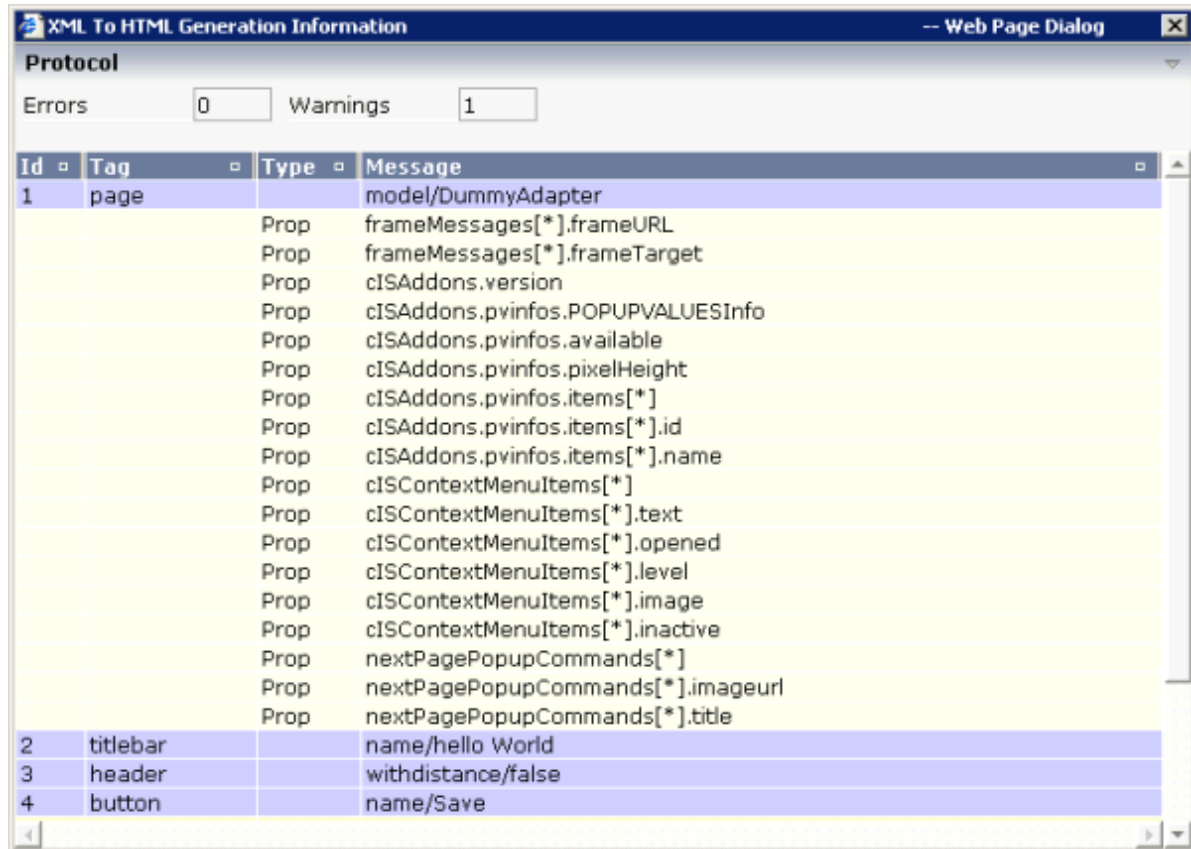
The protocol lists error and warnings for your current layout.

A warning is provided, for example, when a method you have defined in the properties area, has not yet been set. See [Using the Code Assistant](#) for further information.

▶ To view the protocol


- From the **Home** tab of the Layout Painter, choose **Protocol**.

The following dialog appears.



Server Log

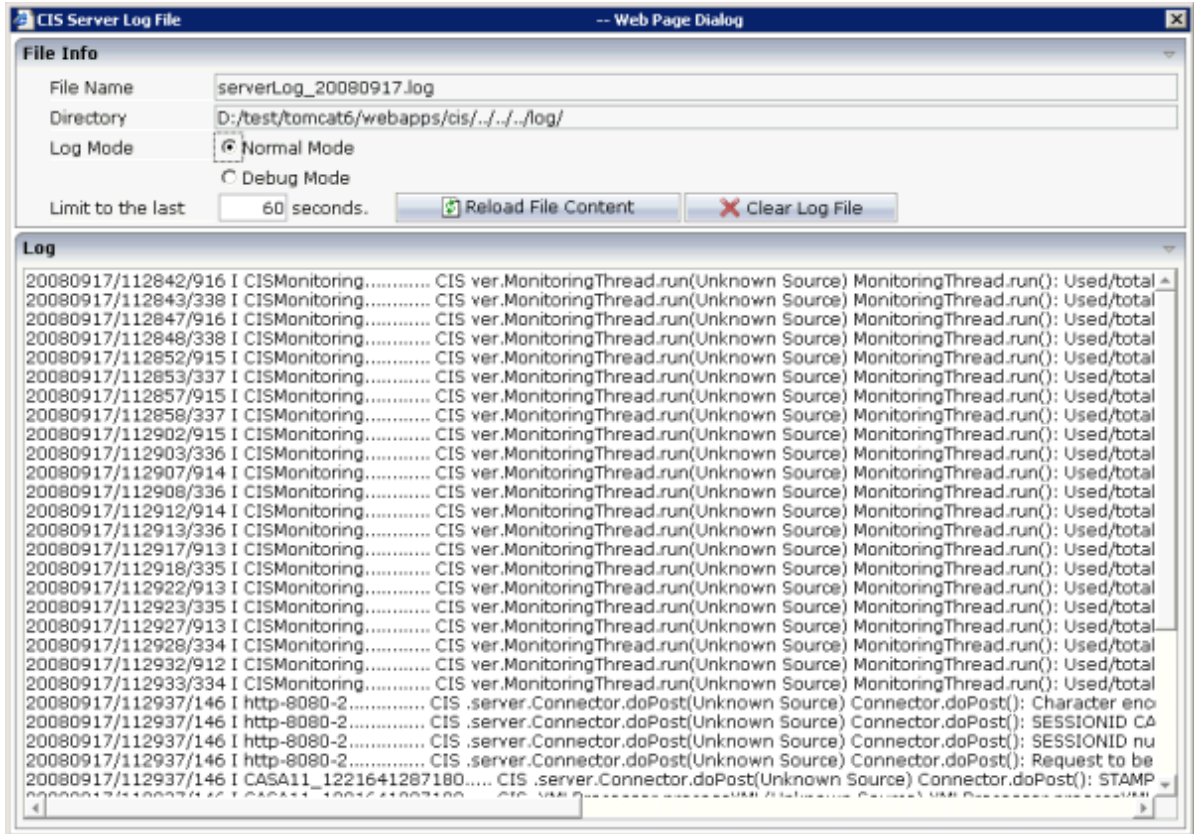
When a layout is not displayed correctly on the client, you can check the data which the server has sent to the client.

 **Note:** This dialog can also be accessed from the [monitoring](#) tool.

▶ To view the server log

- From the **Home** tab of the Layout Painter, choose **Server Log**.

A dialog appears. For example:



The name of the log file and the position at which it is stored is shown at the top of the dialog.

The following options are available:

Log Mode

Choose the option button for the required log mode: **Normal Mode** or **Debug Mode**.

In debug mode, the log contains more detailed information.

Limit to the last n seconds

This option is useful when used together with the **Reload File Content** button. You can specify a different number of seconds.

Reload File Content

When you choose this command button, only the last n seconds (see the above option) are shown in the server log.

Clear Log File

When you choose this command button, the content of the log file is deleted.

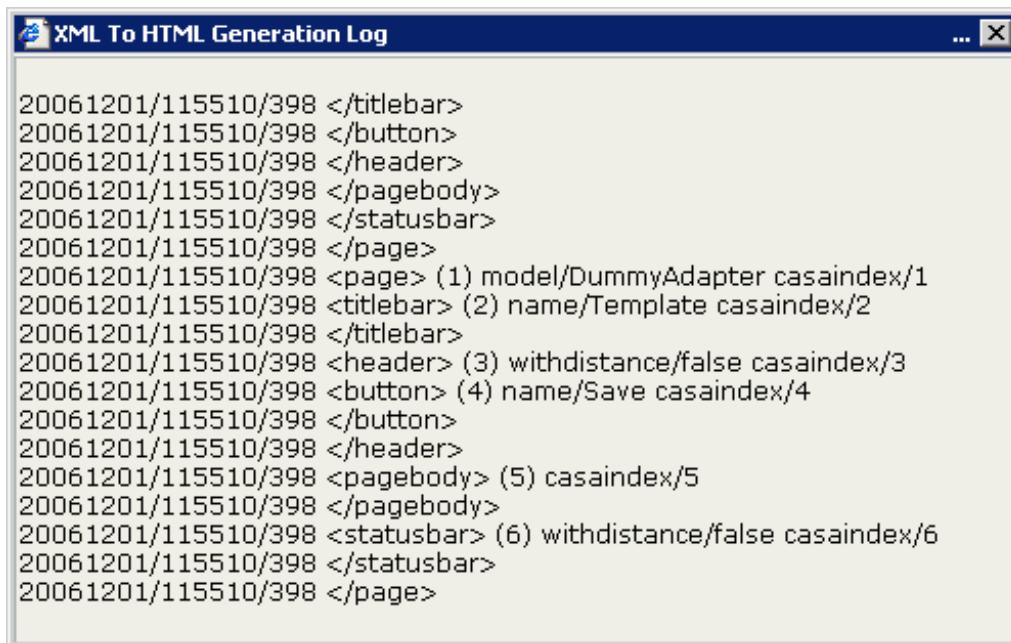
Log

You can display information on the controls that have been written to the generated HTML page.

▶ **To view the log**

- From the **Home** tab of the Layout Painter, choose **Log**.

A dialog appears. For example:



```
20061201/115510/398 </titlebar>
20061201/115510/398 </button>
20061201/115510/398 </header>
20061201/115510/398 </pagebody>
20061201/115510/398 </statusbar>
20061201/115510/398 </page>
20061201/115510/398 <page> (1) model/DummyAdapter casaindex/1
20061201/115510/398 <titlebar> (2) name/Template casaindex/2
20061201/115510/398 </titlebar>
20061201/115510/398 <header> (3) withdistance/false casaindex/3
20061201/115510/398 <button> (4) name/Save casaindex/4
20061201/115510/398 </button>
20061201/115510/398 </header>
20061201/115510/398 <pagebody> (5) casaindex/5
20061201/115510/398 </pagebody>
20061201/115510/398 <statusbar> (6) withdistance/false casaindex/6
20061201/115510/398 </statusbar>
20061201/115510/398 </page>
```

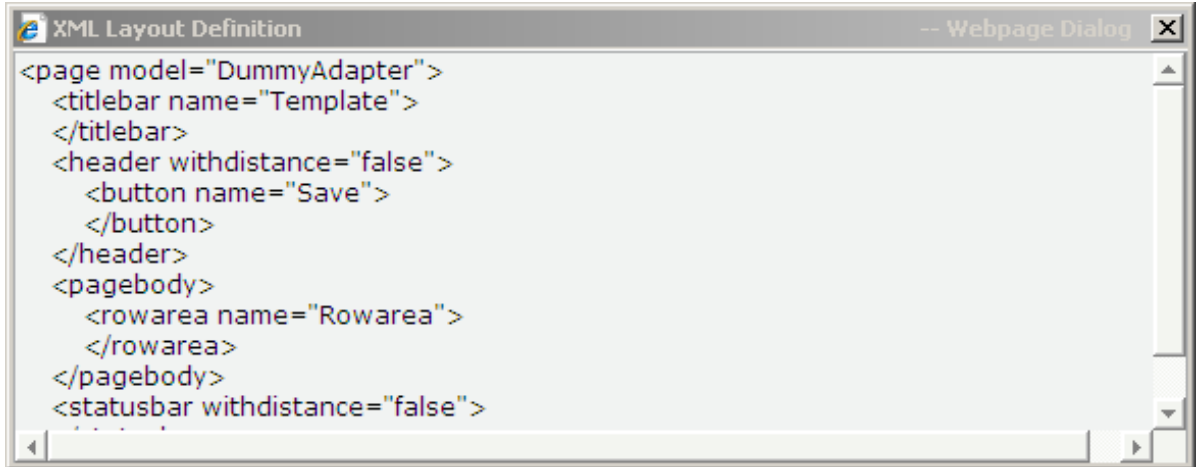
XML

When creating the layout, you can view the currently defined XML code at any point of time.


▶ **To view the XML code**

- From the **Edit** tab of the Layout Painter, choose **XML**.

A dialog appears. For example:



```
<page model="DummyAdapter">
  <titlebar name="Template">
  </titlebar>
  <header withdistance="false">
    <button name="Save">
    </button>
  </header>
  <pagebody>
    <rowarea name="Rowarea">
    </rowarea>
  </pagebody>
  <statusbar withdistance="false">
```

 **Note:** The properties for a tag which you leave blank are not shown in the XML code.

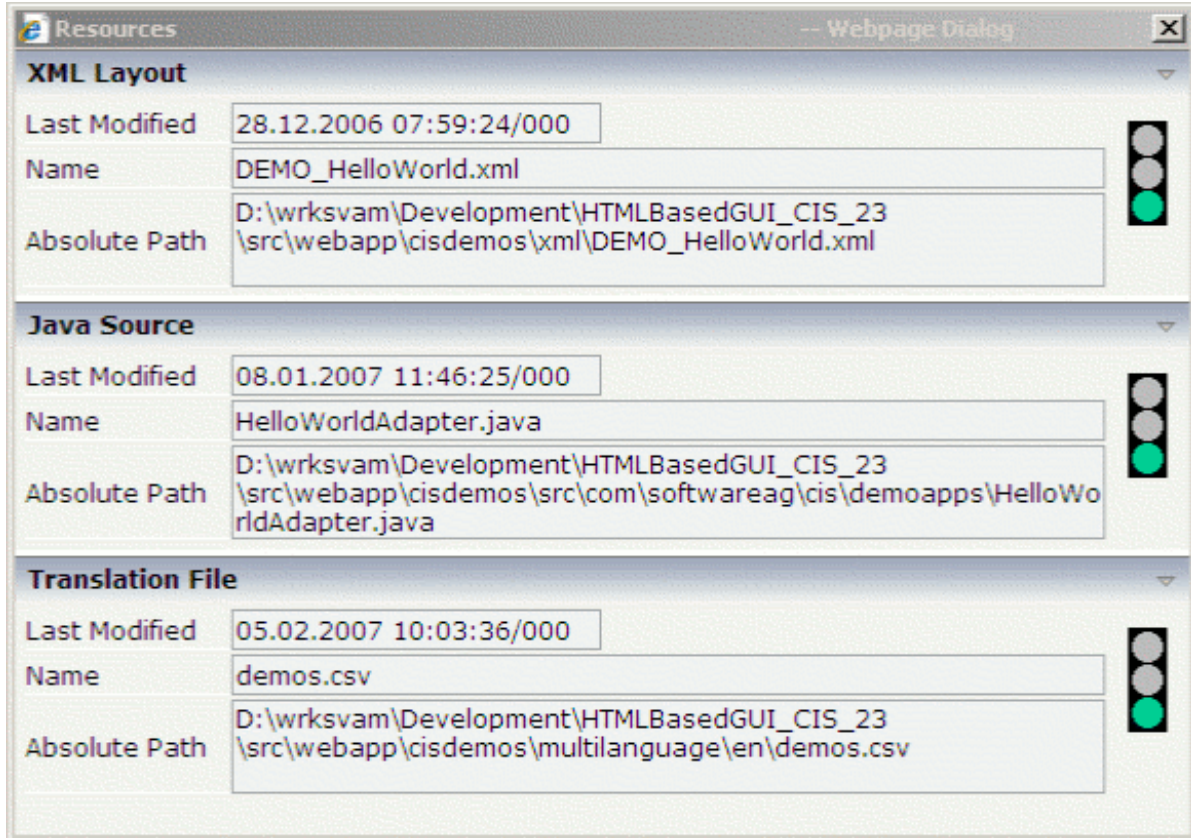
Resources

You can display information on the resources that are used for the current layout. The resources are the XML layout definition, the Java source and the translation file.

▶ To view the resources

- From the **Edit** tab of the Layout Painter, choose **Resources**.

A dialog appears. For example:



The dialog shows the file names for the different resources, the paths to these files, and the date and time when the resources have been modified last.

A traffic light is shown for each resource. One of the following colors can be shown:

Color	Description
Red	The resource cannot be written.
Yellow	Modifications have not yet been saved.
Green	The resource has been saved successfully.

III

■ 13 Layout Manager	89
■ 14 Style Sheet Editor	99
■ 15 Language Manager	111
■ 16 Literal Translator	119
■ 17 WAR Packager	125

13

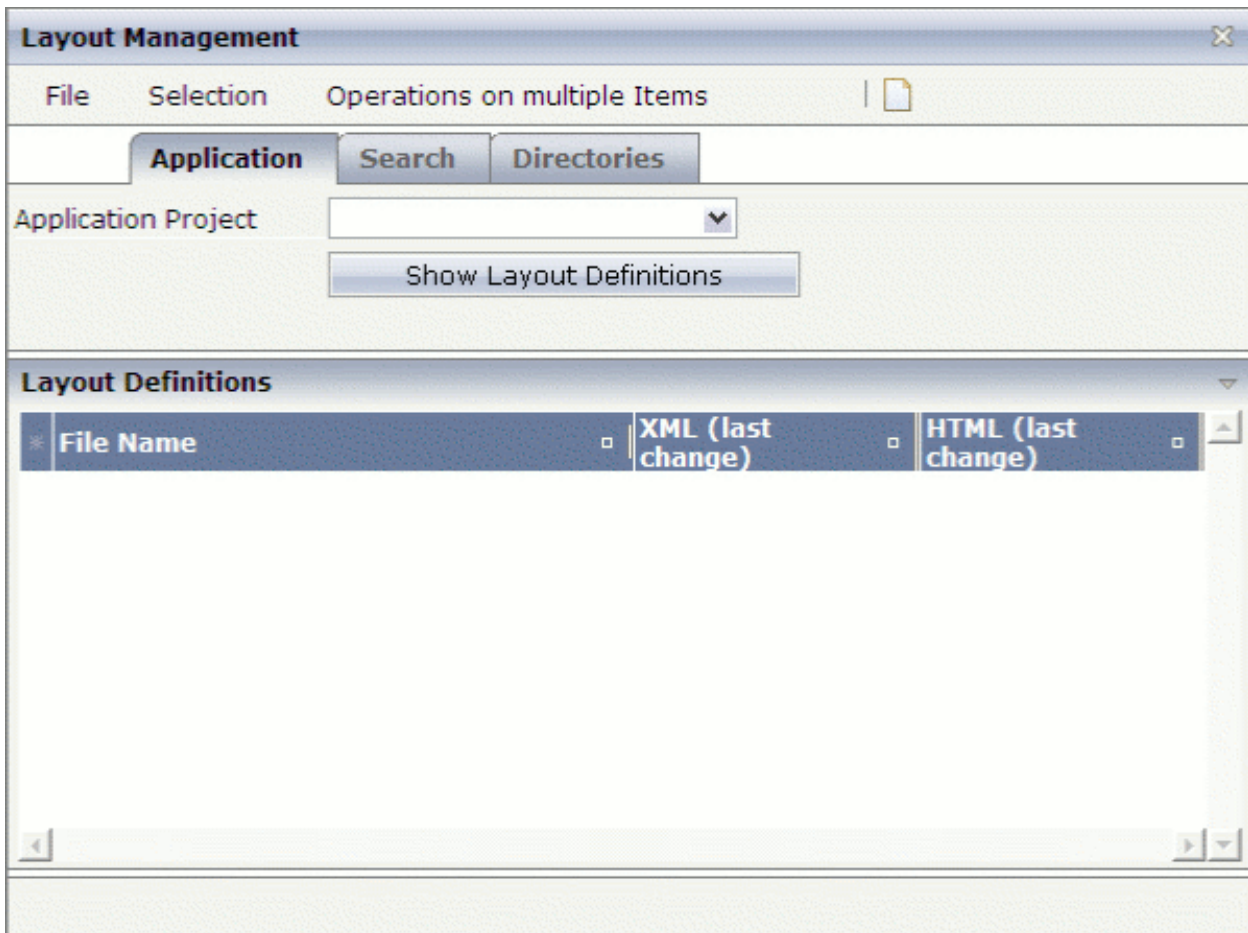
Layout Manager

▪ Invoking the Layout Manager	90
▪ Displaying the Layout Definitions for a Project	91
▪ Searching for Layout Definitions	92
▪ Selecting Layout Definitions	93
▪ Generating HTML Pages	93
▪ Removing HTML and XML Pages	94
▪ Invoking the Layout Painter	95
▪ Directories	96
▪ Java API for HTML Page Mass Regeneration	97

With the Layout Manager you can manage the layouts for your projects. It provides for a mass (re)generation of layout definitions into corresponding HTML pages. You can also invoke the Layout Painter from the Layout Manager (for example, by opening an existing layout or by creating a new layout).

Invoking the Layout Manager

When you invoke the Layout Manager, the following dialog appears. It is initially empty.



▶ **To invoke the Layout Manager**

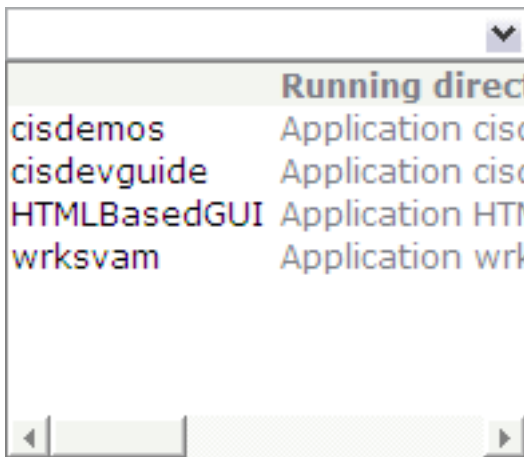
- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Layout Manager**.


Displaying the Layout Definitions for a Project

In the Layout Manager, you first have to specify the project for which you want to manage the layouts.

▶ To display the layout definitions for a project

- From the **Application Project** drop-down list box, select the required project.



 **Note:** Do not use the projects that are delivered with Application Designer. This also includes the first entry in the drop-down list box for which a name is not shown.

The layout definitions for the selected project are shown. For example:

A screenshot of the 'Layout Definitions' window. It displays a table with the following data:

* File Name	XML (last change)	HTML (last change)
hellotranslate.xml	01.12.2006 11:05:08/231	01.12.2006 11:05:08/184
helloworld.xml	29.11.2006 15:12:30/782	29.11.2006 15:12:30/735
ZZZZZZZZGenerated.xml	01.12.2006 12:01:15/989	01.12.2006 12:01:15/942

For each layout definition, the date and time of the last change of the XML file are shown, and (if available) the date and time when the latest HTML page has been generated. In the time, the number after the slash indicates a thousandth of a second.

**Notes:**

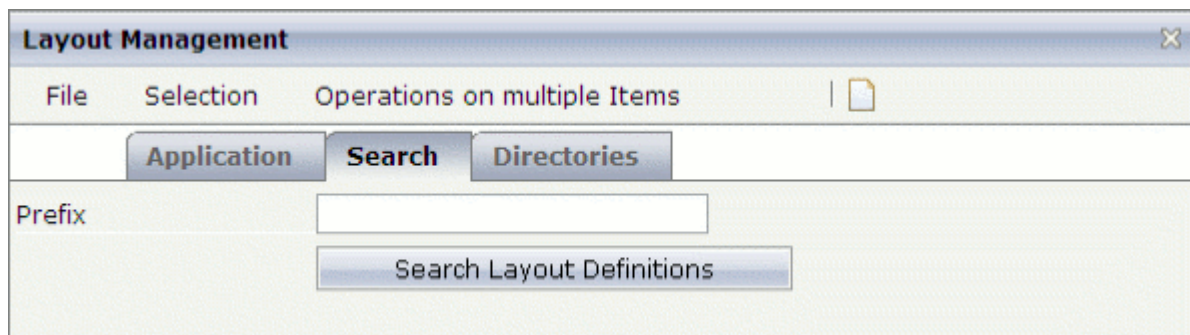
1. For each preview, a file with the name *ZZZZZZZZGenerated.xml* is automatically generated.
2. When you have not selected a specific project and choose the **Show Layout Definitions** button, the layout definitions of all projects are shown.

Searching for Layout Definitions

If there are many layout definitions inside your project, you can limit the number of layout definitions to those starting with specific characters.

▶ To search for layout definitions

- 1 Choose the **Search** tab at the top of the dialog.



- 2 In the **Prefix** text box, enter the first characters of the layout definition(s) that you want to display.



Note: Do not enter a wildcard character such as the asterisk (*).

- 3 Choose the **Search Layout Definitions** button.

If the corresponding layout definitions can be found in the current project, they are listed at the bottom of the dialog.

▶ **To redisplay the layout definitions**

- 1 Choose the **Application** tab at the top of the dialog.
- 2 Choose the **Show Layout Definitions** button.

All layout definitions for the selected project are shown at the bottom of the dialog.

Selecting Layout Definitions

In order to generate HTML pages or to remove HTML and/or XML files (see below), you first have to select the required entries in the **Layout Definitions** area at the bottom of the dialog.

You can select the layout definitions in different ways:

- You can select a single layout definition by selecting the corresponding row.
- You can use the CTRL or SHIFT key to select (or deselect) more than one row.
- You can click the small asterisk (*) at the top left of the table to select (or deselect) all rows.
- You can choose one of the following commands in the **Selection** menu:

Command	Description
Select all	Selects all layout definitions.
Select outdated	Selects all layout definitions for which an HTML page has not yet been generated, or for which the time stamp of the HTML file is earlier than the last time stamp of the corresponding XML file.
Deselect all	Deselects all selected layout definitions.

Generating HTML Pages

For each XML layout definition, you can generate the HTML page. When an HTML page does not yet exist, it is generated. When an HTML page already exists, it is regenerated using the latest information from the XML layout definition.

See also: [Java API for HTML Page Mass Regeneration](#).

▶ **To generate or regenerate HTML pages**

- 1 Select the layout definition(s) as described above.
- 2 From the **Operations on multiple Items** menu, choose **(Re)Generate HTML Pages**.

Removing HTML and XML Pages

You can either remove the HTML page(s) for the selected XML layout definition(s), or you can remove both the XML layout definition(s) and the corresponding HTML page(s).

▶ To remove the HTML pages

- 1 Select the layout definition(s) as described above.
- 2 From the **Operations on multiple Items** menu, choose **Remove HTML Pages**.

For all selected XML layout definitions, the HTML pages are deleted from the file system. You are not asked to confirm the deletion.

▶ To remove the XML layout definition and the corresponding HTML pages

- 1 Select the layout definition(s) as described above.



Note: If you want to choose the command from the context menu (see below), it is only possible to select a single layout definition.

- 2 From the **Operations on multiple Items** menu, choose **Remove XML and HTML Pages**.

Or:

Invoke the context menu and choose the **Remove XML and HTML Pages** command.

A dialog box appears asking whether you really want to remove the selected XML layout definitions.

- 3 Choose the **Yes** button.

All selected XML layout definitions are removed from the system together with their corresponding HTML pages. They are no longer shown in the **Layout Definitions** area at the bottom of the dialog.

Invoking the Layout Painter

The Layout Manager provides different commands for invoking the Layout Painter.

▶ To create a layout in the currently selected project

- 1 From the **File** menu of the Layout Manager, choose **New**.

Or:

Choose the following button from the toolbar of the Layout Manager.



The Layout Painter is invoked and the dialog appears in which you have to enter the name of the file that is to contain your layout definition. See also [Creating a Layout](#) in the description of the Layout Painter.

- 2 Enter a file name and select the template that you want to use.

▶ To open a layout in the Layout Painter

- 1 Select the layout definition in the table at the bottom of the dialog.
- 2 Invoke the context menu and choose the **Open in Layout Painter** command.

The Layout Painter is invoked and the content of the **HTML Preview** tab is shown. See [Defining the XML Layout](#) for further information.

▶ To invoke the Code Assistant

- 1 Select the layout definition in the table at the bottom of the dialog.
- 2 Invoke the context menu and choose the **Generate Adapter Code** command.

The Layout Painter is invoked and the content of the **Code Assistant** tab is shown. See [Using the Code Assistant](#) for further information.



Note: The Code Assistant can only be invoked, if the Java source directory has been defined. See the description of the [Configuration](#) dialog.

▶ To invoke the Literal Assistant

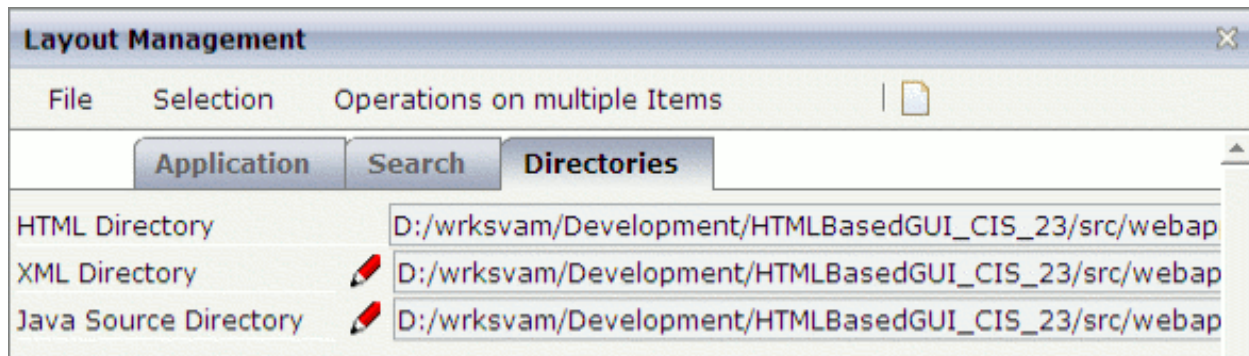
- 1 Select the layout definition in the table at the bottom of the dialog.

- Invoke the context menu and choose the **Maintain Literals** command.

The Layout Painter is invoked and the content of the **Literal Assistant** tab is shown. See [Using the Literal Assistant](#) for further information.

Directories

You can display the paths to the resource directories in the file system.



Information is shown for the following directories:

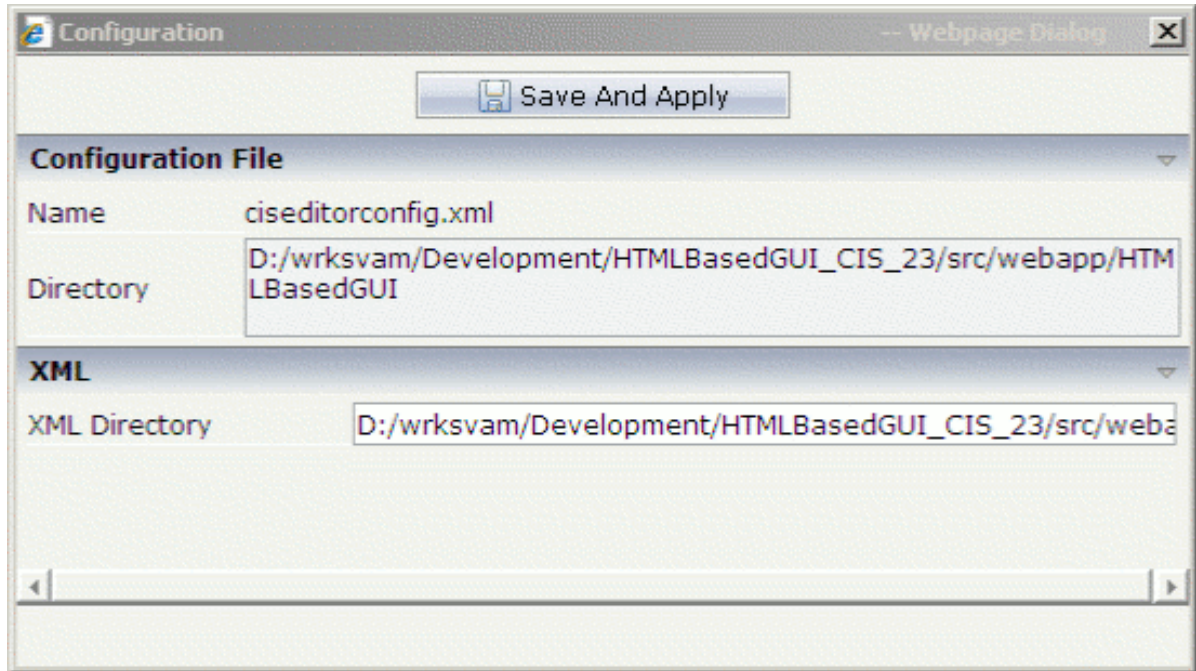
Directory Type	Description
HTML Directory	The HTML pages are always located in the root of your the application project directory. This location cannot be changed.
XML Directory	All development tools read/write the XML layout definitions from/to this directory. By default, this directory is set to <i><your-project>/xml</i> . If you do not want to have the layouts within your application project, you can define another directory somewhere outside of your project (see below).
Java Source Directory	Shows the Java source directory of the currently selected project. By default, this directory is not defined. If you want to use the Code Assistant, you must first define the Java source directory (see below).

▶ To define a different XML or Java source directory

- Choose the **Directories** tab at the top of the dialog.
- Choose the pencil which is shown to the left of the corresponding text box.



A **Configuration** dialog box appears for the selected type of directory. Example for the XML directory:



- 3 Specify the path to the directory.
- 4 Choose the **Save and Apply** button.

Java API for HTML Page Mass Regeneration

Application Designer provides a Java API for regenerating all pages of a given application project. Call the method `main` of class `HTMLGeneratorWholeDirectory` (package `com.softwareag.cis.gui.generate`) with the following argument list:

Argument	Description
1. XML Files Directory	Directory in which the XML layout files are kept.
2. HTML Files Directory	Directory in which the generated HTML pages are to be output.
3. Log Directory	Directory in which the log files are to be output.
4. Accesspath Directory	Directory in which the generated Accesspath files are to be output.

For more details, see the Java API documentation (JavaDoc).

14

Style Sheet Editor

▪ Invoking the Style Sheet Editor	100
▪ Defining the Location for your Own Style Sheets	101
▪ Creating a New Style Sheet	103
▪ Opening an Existing Style Sheet	104
▪ Changing a Style Sheet	105
▪ Overview of Variables	107
▪ Regenerating Your Own Style Sheet from the Style Sheet Template	108
▪ Using a Version Control Tool	109

All controls that are contained inside the Application Designer control library are rendered using a style sheet. Application Designer delivers a variety of predefined styles but also allows to you to create your own styles sheets.

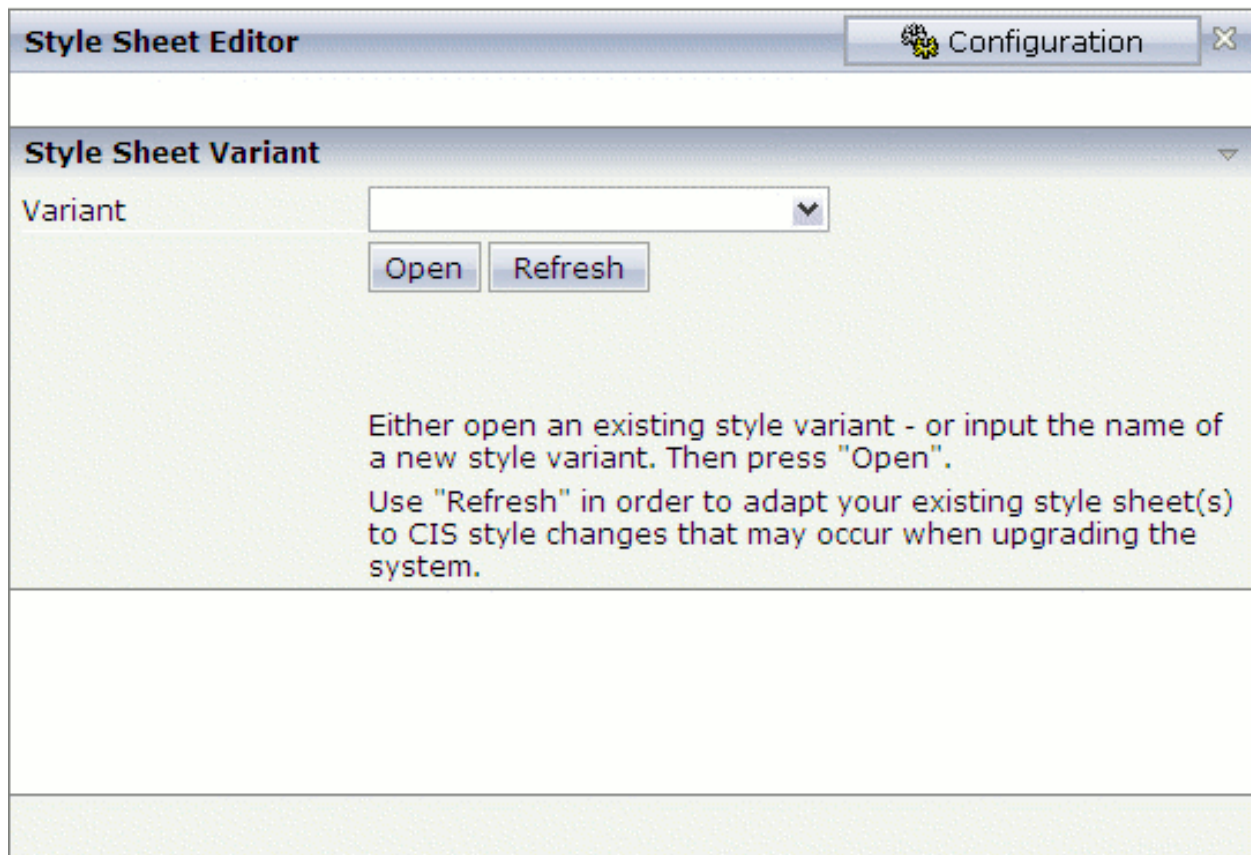
The Style Sheet Editor simplifies the creation of your own style sheets: on the one hand, you can define the very basic style elements (main colors to be used), and on the other hand, you can change a controls' style definition on the lowest level.

For information on how to define a style sheet in the Layout Painter, see [Defining a Style Sheet](#).

For further information on style sheets, see *Adapting the Look & Feel* in the *Special Development Topics*.

Invoking the Style Sheet Editor

When you invoke the Style Sheet Editor, the following dialog appears.



▶ **To invoke the Style Sheet Editor**

- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Style Sheet Editor**.

Defining the Location for your Own Style Sheets

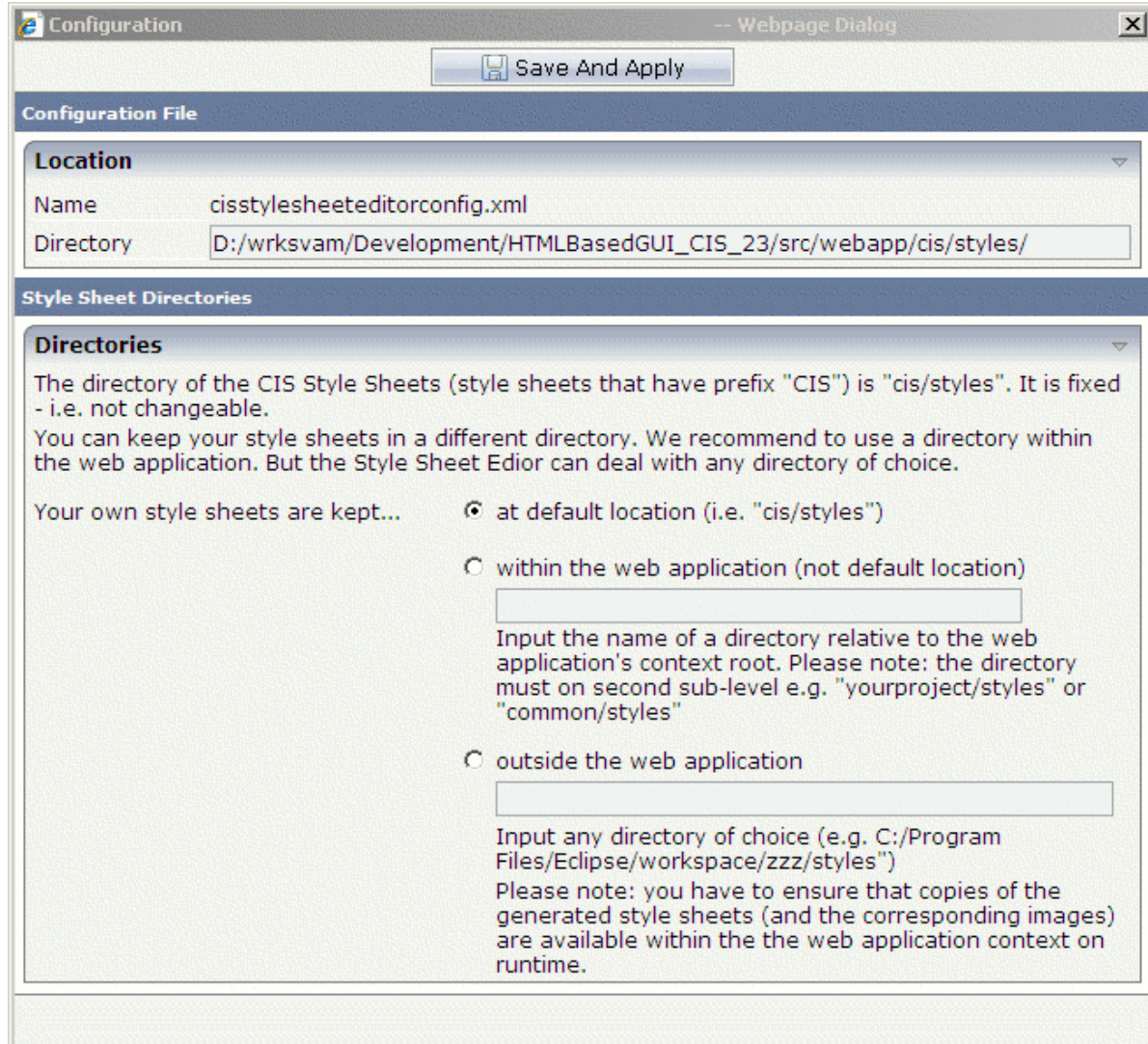
Application Designer provides a set of predefined style sheets. They are available in the directory `<your-webapplication>/cis/styles` and their names have the prefix "CIS_".

When you create your own style sheets, they are placed in the above directory by default. You can also define another directory for your own style sheets.

▶ **To define another location for your own style sheets**

- 1 Choose the **Configuration** button which is shown in top right of the Style Sheet Editor.

The following dialog appears.



2 Choose the option button for the desired location (within the web application or outside the web application).

3 Specify the path to the directory.


If you want to store your own style sheets within the web application (which is recommended), you have to specify a relative path.

4 Choose the **Save and Apply** button.

Creating a New Style Sheet

When you create and save a new style sheet, it is written to the directory that has been defined in the [Configuration](#) dialog.

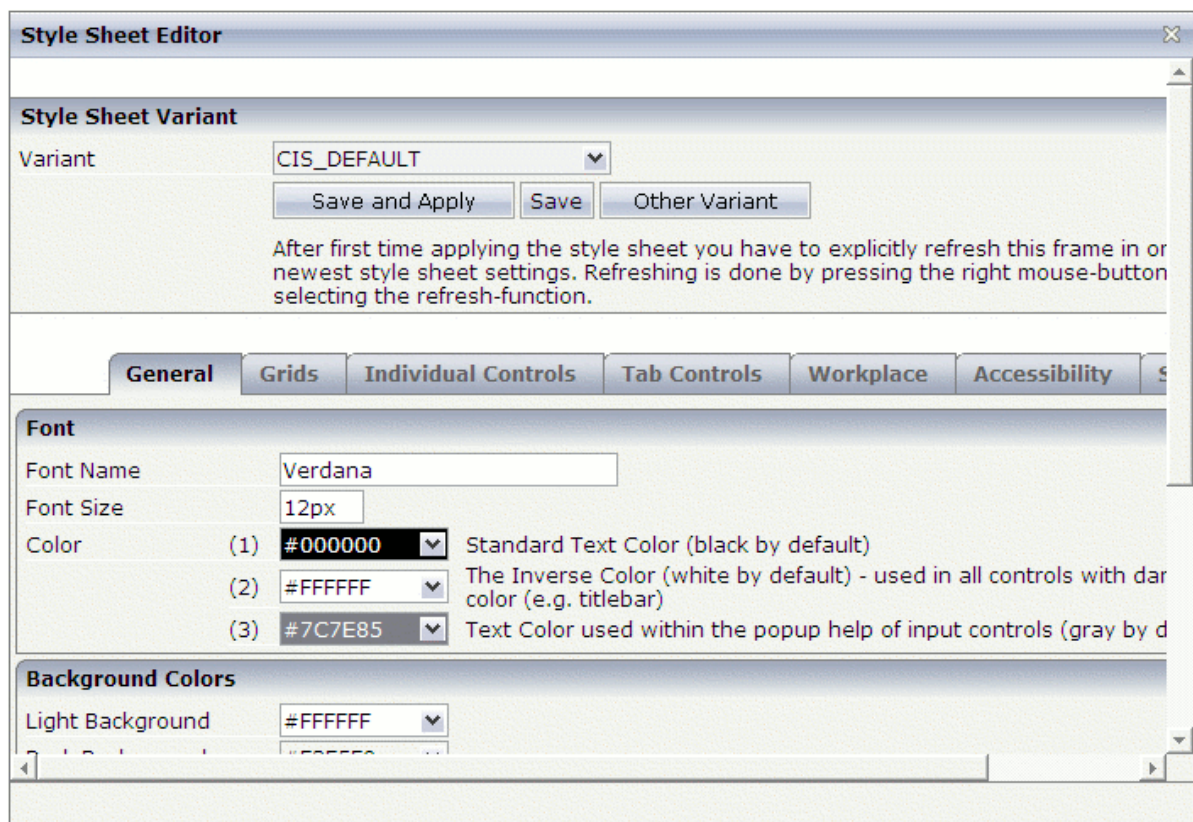
You can determine that your new style sheet is immediately applied to the current session.

 **Note:** By default, your pages are rendered with the *CIS_DEFAULT* style sheet which is one of Application Designer's predefined style sheets.

▶ To create a new style sheet

- 1 In the **Variant** drop-down list box, enter the name for your new style sheet.
- 2 Choose the **Open** button.

The following dialog appears.



- 3 Modify the values on the different tabs according to your requirements. See [Changing a Style Sheet](#) for information on the **Style Details** tab.



Note: When you choose the **Other Variant** button, the dialog is shown again which appears when you invoke the Style Sheet Editor. You can then open another style sheet or create a new style sheet. If you have not saved your input previously, it is lost.

- 4 To save your changes, choose either the **Save** button or the **Save and Apply** button.

When you save a new style sheet for the first time, it is created in your style sheet directory with the extension `css`.

When you choose the **Save and Apply** button, the style sheet file is immediately applied to the current session.

Opening an Existing Style Sheet

When you open an existing style sheet, you can modify it according to your requirements and/or immediately apply the style sheet to the current session.

▶ To open an existing style sheet

- 1 From the **Variant** drop-down list box, choose the name of the style sheet that you want to open.



Note: In addition to the style sheet names, the drop-down list box also shows the paths to the corresponding files in the file system.

- 2 Choose the **Open** button.

The dialog with the style sheet definitions appears.

- 3 Modify the values on the different tabs according to your requirements. See [Changing a Style Sheet](#) for information on the **Style Details** tab.



Note: When you choose the **Other Variant** button, the dialog is shown again which appears when you invoke the Style Sheet Editor. You can then open another style sheet or create a new style sheet. Any changes that you have applied after the last save are lost.

- 4 To save your changes, choose either the **Save** button or the **Save and Apply** button.

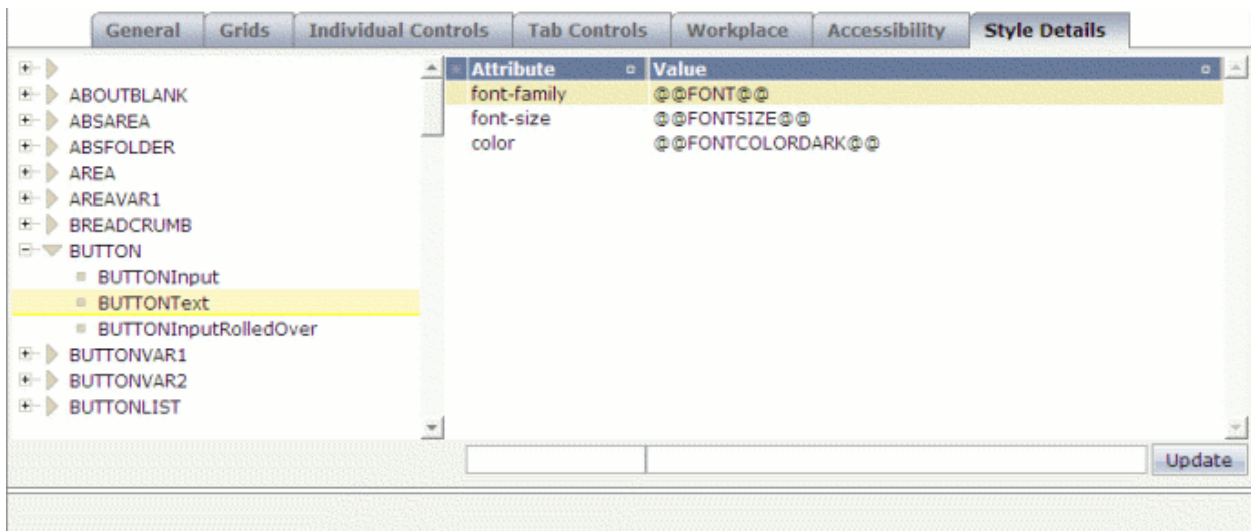
When you choose the **Save and Apply** button, the style sheet file is immediately applied to the current session.

Changing a Style Sheet

Application Designer provides an internal style template in which all the default style information is kept. A newly created style sheet is identical to this template. All style updates that you apply to your style sheet are stored within a separate file which has the extension *info*. For example, if you named your style sheet "MyStyle", your style sheet directory contains the file *MyStyle.info*.

 **Note:** The style sheet directory also contains GIF files. They are generated according to your specifications for background, font, border color, etc. Each style sheet has thus individual GIF files. When you modify, for example, a color setting, new GIF files are generated which overwrite the previous GIF files.

When you edit a style sheet, you define the standard settings such as the font size on the **General** tab. All of the settings defined on the **General** tab are then assigned to individual controls on the **Style Details** tab via the corresponding variable (for example, @@FONTSIZE@@). See [Overview of Variables](#).



▶ To change a style sheet

- 1 Expand the tree on the left and select the name of a style sheet class.

The attributes for the selected class are shown on the right.

- 2 Select the attribute that you want to change.

An arrow is shown at the beginning of the selected line. The attribute name and value are now shown in the text boxes at the bottom of the dialog.

- 3 Specify another value for the attribute. For example, if you want to define another color, you can define a color value such as #FF0000 (instead of @@FONTCOLORDARK@@).

You can also define a different name for a property. As soon as you choose the **Update** button, a new entry is created in the style sheet. This entry must be a valid CSS style sheet definition. For example, when you change the attribute name from `color` to `background-color`, this is a valid definition. When you specify an invalid attribute name, this will not have any effect. For example, when you change the attribute name from `color` to `hello`, this will not have any effect even though the new entry `hello @@FONTCOLORDARK@@` will be created.

- 4 Choose the **Update** button.

When you have changed a default entry of the Application Designer style template, the changed information is now shown in bold in an additional line. For example:

* Attribute	Value
color	#FFFF00
font-family	@@FONT@@
font-size	@@FONTSIZE@@
color	@@FONTCOLORDARK@@

 **Notes:**

1. If you do not want to take over a property from the style sheet template, specify "DELETE" as the attribute value and choose the **Update** button. All attributes that are marked with "DELETE" are not included in your style sheet.
2. If you want to take back a style update (that is: a change which is indicated by a bold line), delete the corresponding attribute value (that is: set it to blank) and choose the **Update** button. The bold line will disappear.

Overview of Variables

The following variables can be defined on the [Style Details](#) tab:

@@FONTCOLORDARK@@
@@FONTSIZE@@
@@FONT@@
@@LIGHTBACKGROUND@@
@@HEADLINEBACKGROUND@@
@@BORDERCOLOR@@
@@DARKBACKGROUND@@
@@TITLEBARBACKGROUND@@
@@FONTCOLORLIGHT@@
@@BUTTONHEIGHT@@
@@BUTTONIMAGE@@
@@BUTTONCOLOR@@
@@FONTCOLORINACTIVE@@
@@CONTROLERRORBACKGROUND@@
@@CONTROLEDITBACKGROUND@@
@@CONTROLDISPLAYBACKGROUND@@
@@LIGHTTITLEBARBACKGROUND@@
@@VARIANT@@
@@FIELDHEIGHT@@
@@FIELDBORDERCOLOR@@
@@CONTROLPOPUPINPUTONLYBACKGROUND@@
@@SHADED DARKBACKGROUND@@
@@SELECTEDCELLBACKGROUND@@
@@SELECTEDBACKGROUND@@
@@EVENCELLBACKGROUND@@
@@ODDCELLBACKGROUND@@
@@TABAREALEFTPADDINGFIRST@@
@@TABAREALEFTPADDINGSECOND@@
@@EMPTYCELLBACKGROUND@@
@@SHADEDSELECTEDBACKGROUND@@
@@ODDCELLBACKGROUNDVAR1@@
@@EVENCELLBACKGROUNDVAR1@@
@@SELECTEDCELLBACKGROUNDVAR1@@
@@EMPTYCELLBACKGROUNDVAR1@@
@@ODDCELLBACKGROUNDVAR2@@
@@EVENCELLBACKGROUNDVAR2@@
@@SELECTEDCELLBACKGROUNDVAR2@@
@@EMPTYCELLBACKGROUNDVAR2@@
@@TITLEBARHEIGHT@@

@@COLORTOPIC1@@
@@COLORTOPIC2@@
@@COLORTOPIC3@@
@@COLORTOPIC4@@
@@COLORTOPIC5@@
@@COLORTOPIC6@@
@@COLORTOPIC7@@
@@COLORTOPIC8@@
@@COLORTOPIC9@@
@@COLORTOPIC10@@

Regenerating Your Own Style Sheet from the Style Sheet Template

From release to release, Application Designer adds new controls to its control library. As a consequence, the style sheet template is typically enhanced with every new control (for example, new style classes are added). As your style changes are kept in a separate file which has the extension *info*, your style sheet can easily include the enhancements. You just have to regenerate your own style sheet file.



Important: After the installation of a new Application Designer build, you have to regenerate your own style sheet(s).

▶ To regenerate your own style sheet

- 1 From the **Variant** drop-down list box, select the name of your style sheet.
- 2 Choose the **Refresh** button.

The following command buttons are now shown: **Selected Variant** and **All Variants**.

- 3 Choose the **Selected Variant** button.



Note: Choose the **All Variants** button if you want to regenerate all existing style sheets.

Using a Version Control Tool

If you want to secure your style sheets with a version control tool, it is sufficient to keep the files with the extension *info* under version control. The actual style sheet file (with the extension *css*) is just a generation result out of the INFO file.

15 Language Manager

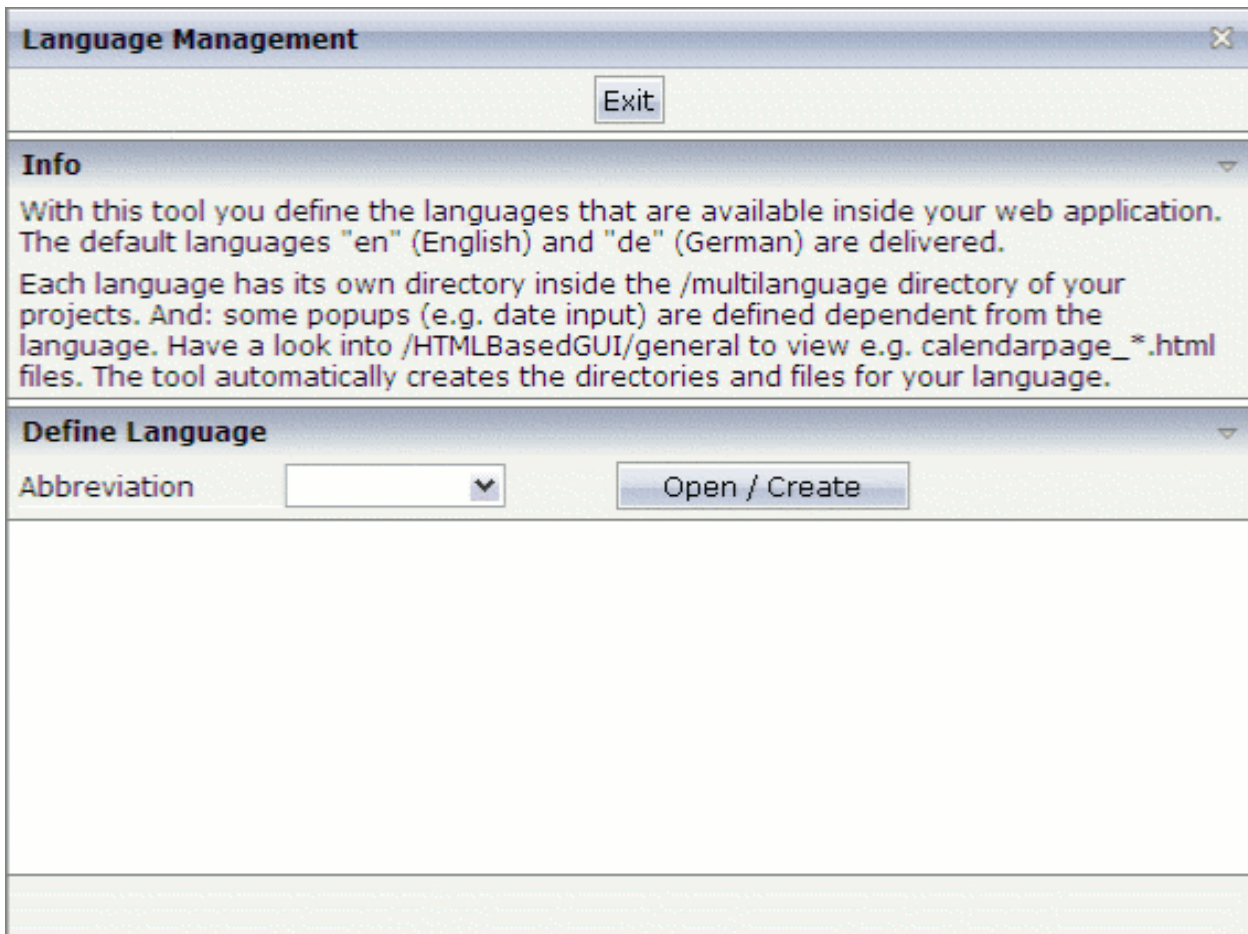
- Invoking the Language Manager 112
- Defining a New Language 113
- Opening an Existing Language 116

Application Designer comes with two languages: "en" for English and "de" for German. The Language Manager can be used to define additional languages.

The concept of the multi language management is described in *Multi Language Management*. It is recommended that you read this information before you proceed with the information below.

Invoking the Language Manager

When you invoke the Language Manager, the following dialog appears.



▶ To invoke the Language Manager

- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Language Manager**.

Defining a New Language

When you define a new language, the required directories and certain files holding textual information will automatically be created in all existing projects.

▶ **To define a new language**

- 1 In the text box of the **Abbreviation** drop-down list box, enter the abbreviation for your new language.
- 2 Choose the **Open / Create** button.

The following dialog appears. The language abbreviation that you have specified is shown at the top of the dialog.

Language Management ✕

Info ▶

Define Language ▼

Abbreviation

Translate the literals and message texts listed below and press "Save" afterwards.

Literals **Messages**

Date Input	<input type="text"/>	Today	<input type="text"/>
Time Input	<input type="text"/>	Hour	<input type="text"/>
Text Input	<input type="text"/>	Minute	<input type="text"/>
Number Input	<input type="text"/>	Second	<input type="text"/>
OK	<input type="text"/>	File Upload	<input type="text"/>
		Upload	<input type="text"/>
January	<input type="text"/>	Mo	<input type="text"/>
February	<input type="text"/>	Tu	<input type="text"/>
March	<input type="text"/>	We	<input type="text"/>
April	<input type="text"/>	Th	<input type="text"/>
May	<input type="text"/>	Fri	<input type="text"/>
June	<input type="text"/>	Sa	<input type="text"/>
July	<input type="text"/>	Su	<input type="text"/>
August	<input type="text"/>	Ctrl	<input type="text"/>
September	<input type="text"/>	Alt	<input type="text"/>
October	<input type="text"/>	Shift	<input type="text"/>
November	<input type="text"/>	No match for	<input type="text"/>
December	<input type="text"/>		

- 3 Translate all literals that are shown on the **Literals** tab.
- 4 Choose the **Messages** tab.



- 5 Translate all strings that are shown on the **Messages** tab.

Each "\n" in a string stands for a line break.

"REPLACE" is a placeholder for a variable. It must not be deleted. During runtime, the corresponding value will be used. Example:

Language 1: \nHint for input: REPLACE.\n\n\n
Language 2: \nHinweis für die Eingabe: REPLACE\n\n\n

- 6 Choose the **Save Language** button.

The directories and files for the specified language abbreviation are created. A message appears in the status bar of the Language Manager.

- 7 If you want to find out which directories and files were created, click the message in the status bar.

A dialog appears. Example:



Opening an Existing Language

You can modify the text for literals and messages that you have specified when you have defined a new language (see above).

▶ To open an existing language

- 1 In the text box of the **Abbreviation** drop-down list box, enter the abbreviation of an existing language.
- 2 Choose the **Open / Create** button.

A dialog appears showing the currently defined texts for the literals and messages.

- 3 Edit the required texts on the **Literals** and/or **Messages** tab.

- 4 Choose the **Save Language** button.

The texts are written to the files which have been created for the specified language abbreviation. A message appears in the status bar of the Language Manager.


- 5 If you want to find out which files were affected, click the message in the status bar to display a dialog.

16

Literal Translator

▪ Invoking the Literal Translator	120
▪ Loading the Literals of a Layout	121
▪ Editing the Literals	122
▪ Adding a New Text ID	123
▪ Removing Text IDs	123

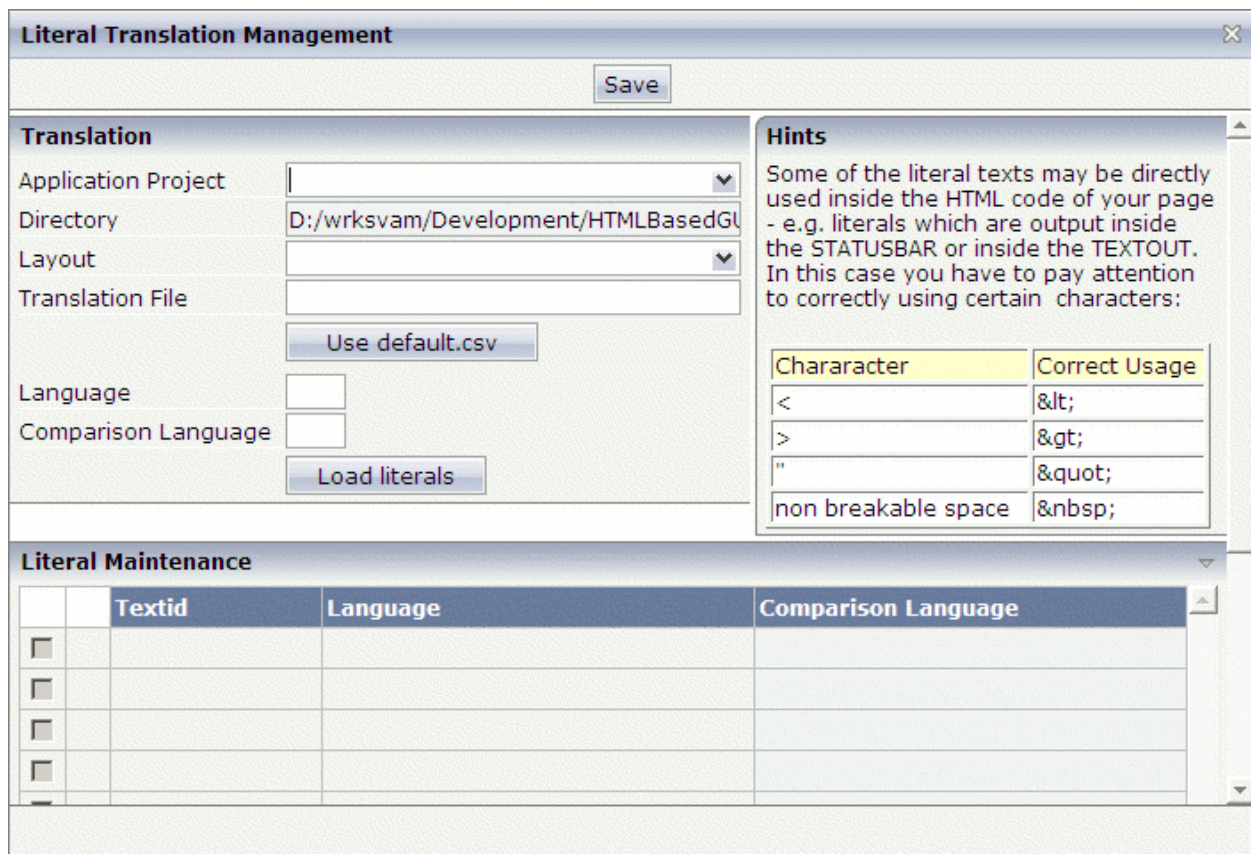
The Literal Translator can be used to translate text IDs. You can also use it to add and remove text IDs.

 **Note:** Text IDs can also be translated using the Literal Assistant which is part of the Layout Painter.

The concept of the multi language management is described in *Multi Language Management*. It is recommended that you read this information before you proceed with the information below.

Invoking the Literal Translator

When you invoke the Literal Translator, the following dialog appears.



▶ **To invoke the Literal Translator**

- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Literal Translator**.

Loading the Literals of a Layout

You can load and modify the texts for a specific layout in one language. Optionally, can also load the literals of a second language for comparison purposes.



Note: If you want to open the literals for a different layout or project, you have to close the Literal Translator and start it once more. Otherwise, the required options are not enabled.

▶ To load the literals of a layout

- 1 In the text box of the **Application Project** drop-down list box, enter the name of the project which contains your layout.

Or:

Open the drop-down list box and choose the project from the resulting dialog.

- 2 In the text box of the **Layout** drop-down list box, enter the name of the layout for which you want to translate the literals.

Or:

Open the drop-down list box and choose the layout from the resulting dialog.

- 3 In the **Translation File** text box, enter the name and extension of your translation file.



Notes:

1. It is not required that the specified translation file already exists. You can **add** new text IDs as described below. When you choose the **Save** button, the translation file will be created and all of your definitions are written to it. Literals that have been created in this way, still have to be assigned to the corresponding controls, for example, using the **Literal Assistant**.
2. If you do not enter the name of a translation file, *default.csv* is automatically provided in this text box.
3. You can use the **Use default.csv** button to enter *default.csv* in this text box.
4. In the **Language** text box, enter the abbreviation for the language that you want to use.
See also *Language Manager*.
5. Optional. If you want to display the literal of a second language for comparison purposes, enter the abbreviation for this language in the **Comparison Language** text box.
6. Choose the **Load literals** button.

The contents of the translation file(s) are loaded and the literals are shown at the bottom of the dialog. Example:

	Textid	Language	Comparison Language
<input type="checkbox"/>	inarea	Input Area	Eingabebereich
<input type="checkbox"/>	input	Input your name and press the 'Say H	Geben Sie Ihren Namen ein und wähle
<input type="checkbox"/>	outarea	Output Area	Ausgabebereich
<input type="checkbox"/>	result	Result	Ergebnis

Loaded literals

Editing the Literals

You can edit the literals that are shown in the **Language** column. It is not possible to edit the literals of the comparison language.

▶ To edit the literals

- 1 Enter the new text directly in the **Language** column.

Or:

To edit a literal in a separate dialog, choose the following icon which is shown next to the literal (this is helpful with long literals):



If you have opened a separate dialog, you have to choose the **Take Over** button in order to confirm your changes and to close the dialog.

- 2 Choose the **Save** button to write your changes to the translation file.

Adding a New Text ID

You can add a new text ID to the translation file and specify a literal for it.

▶ **To add a new text ID**

- 1 Choose the **New Text ID** button which is shown below the list of literals.
An empty line is added to the list.
- 2 Enter the literal in the **Language** column.
- 3 Choose the **Save** button to write the new text ID and literal to the translation file.

Removing Text IDs

You can remove one or more text IDs from the translation file.

▶ **To remove text IDs**

- 1 Select the check boxes in front of the text IDs that you want to remove.
- 2 Choose the **Remove Selected Text ID(s)** button which is shown below the list of literals.
- 3 Choose the **Save** button to remove the text IDs from the translation file.

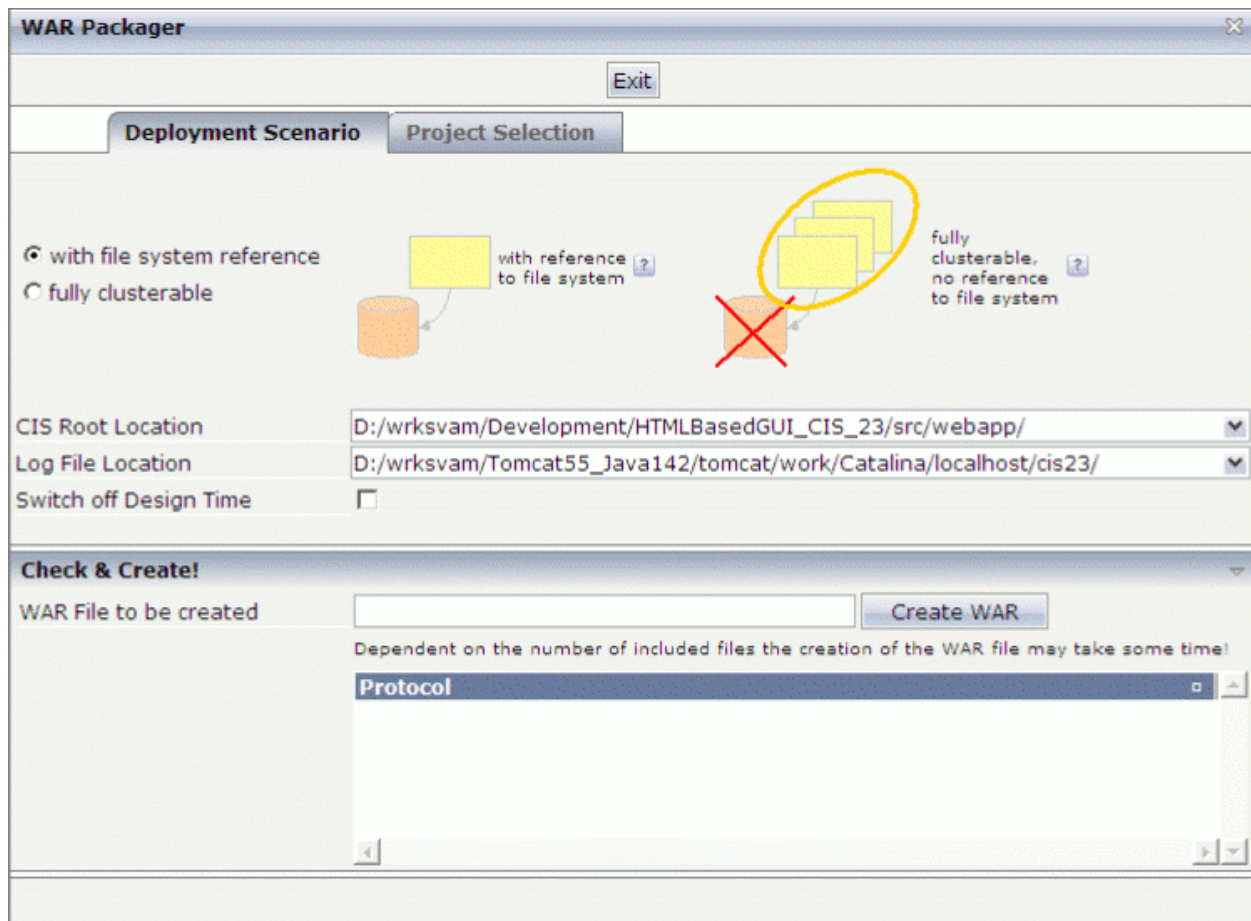
17 WAR Packager

- Invoking the WAR Packager 126
- Types of Generation 127
- Creating a Web Archive 128
- Java API for WAR File Generation 129

In order to deliver the web applications that you have built with Application Designer, you need to create a web archive (WAR file). The WAR Packager provides an easy-to-use graphical user interface that you can use to create the web archive.

Invoking the WAR Packager

When you invoke the WAR Packager, the following dialog appears.



▶ To invoke the WAR Packager

- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **WAR Packager**.

Types of Generation

When you create a web archive, you have to define the type of generation:

- [Web Archive with Reference to the File System](#)
- [Web Archive without Reference to the File System](#)

Web Archive with Reference to the File System

This is the standard mode in which Application Designer is delivered. The name of the corresponding option in the WAR Packager is **with file system reference**.

In the web archive's *web.xml* file there are references to the web application's root in the file system and to the location where log files are written. The development functions of Application Designer (such as the Layout Painter) require access to the file system and as a consequence know where to read and write files. In addition, the Application Designer class loader, which simplifies development, accesses application classes through the file system (*appclasses* directory of a project).

This mode is ideal for development but has some disadvantages in clustered runtime scenarios:

- With some applications servers, you do not know the location of the deployed files (or: you should not know the location at all).
- In clustered scenarios, you would require to have one directory that is the same for all cluster nodes.

Web Archive without Reference to the File System

This is the preferred deployment mode for applications. The name of the corresponding option in the WAR Packager is **fully clusterable**.

The resources are not read from the file system but are read via the servlet engine's web resource reader. The log is written into the log of the servlet engine as well. Classes are not read via the Application Designer class loader but are read from the directory *WEB-INF/classes* or *WEB-INF/lib*.

There is nothing to be aware of from the Application Designer side when running the application in an application server cluster.

When creating the web archive in this mode, the following is implicitly done:

- The file *cisconfig.xml* is adapted so that it does not use the Application Designer class loader and does not to start a monitoring thread of its own.
- All class files in the directory *<project>/appclasses/classes* are copied to *WEB-INF/classes*.
- All files in the directory *<project>/appclasses/lib* are copied to *WEB-INF/lib*.

Creating a Web Archive

The WAR file is written to the *bin* directory of your servlet container. If you are running a standard installation of Application Designer, the file is written to `<installdir>/tomcat/bin`.

See also: [Java API for WAR File Generation](#).

► To create a web archive

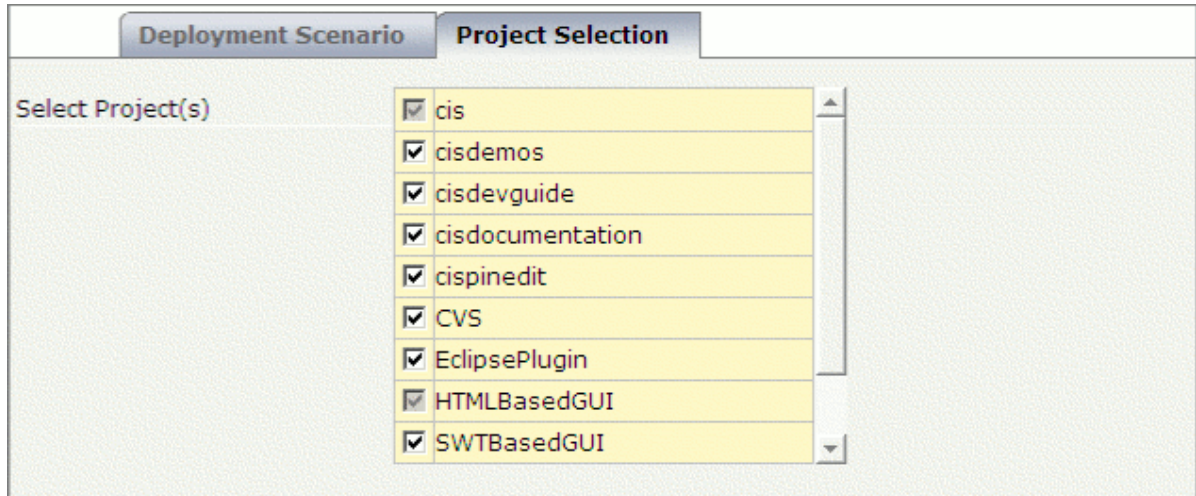
- 1 Define the generation type by selecting one of the following option buttons: **with file system reference** or **fully clusterable**.
- 2 When you have selected the option button **with file system reference**, you have to specify the following:

Option	Description
CIS Root Location	<p>The directory which contains your web applications.</p> <p>You can also select <code>REALPATH</code> from the drop-down list box. This variable contains the servlet engine's path to the web applications. It is filled during runtime.</p> <p>Caution: <code>REALPATH</code> is not supported by all servlet engines.</p>
Log File Location	<p>The directory which contains your log files.</p> <p>You can also select one of the following from the drop-down list box:</p> <ul style="list-style-type: none"> ■ <code>REALPATH/xxx</code> The log files are written to a subdirectory of the <code>REALPATH</code> (see the above description), where <code>xxx</code> stands for the name of the subdirectory. ■ <code>TEMP</code> This variable contains the servlet engine's path to the temporary directory. It is filled during runtime. The <code>TEMP</code> variable is supported by all servlet engines.
Switch off Design Time	Optional. Select this check box if you do not want to include the development tools and Application Designer's documentation in your web archive.



Note: The above options are disabled if you have selected the option button **fully clusterable**.

- 3 Select the **Project Selection** tab.



- 4 Select the application projects that you want to include in your web archive.

The project "HTMLBasedGUI", which contains the Application Designer resources, cannot be deselected. The projects starting with "CIS" contain demos only; if you want, you can deselect these projects.

- 5 In the text box **WAR File to be created**, enter the name and extension for your web archive file.
- 6 Choose the **Create WAR** button.

The web archive is created.

After the creation, a protocol is available in the lower part of the WAR Packager dialog.

Java API for WAR File Generation

Application Designer provides a Java API for generating a WAR file. Call the method `main` of class `WARGenerator` (package `com.softwareag.cis.editor`) with the following argument list:

Argument	Description
1. Application Designer home directory	Mandatory. The directory which contains your web applications.
2. Temporary directory	Mandatory. A temporary directory.
3. Result file name	Mandatory. The name of the generated WAR file.
4. Clusterable indicator	Mandatory. A flag (<code>true</code> or <code>false</code>) which indicates whether the generated WAR file can be used within a server cluster.

Argument	Description
5. Switch off design time	Mandatory. A flag (<code>true</code> or <code>false</code>) which indicates whether the generated WAR file contains the Application Designer design time resources (tools).
6. Project name	Optional. The name of the application project that is to be part of the WAR file. If you want to add multiple projects, add further names subsequently.

For more details, see the Java API documentation (JavaDoc).

IV Control Editor

You use the Control Editor to build your own custom controls.

The description of the Control Editor is organized under the following headings:

[Using the Control Editor](#)

[Defining a Control](#)

[Examples](#)



Note: Before you proceed with the information in this documentation, it is recommended that you first become familiar with the control development with Application Designer. For detailed information, see the *Custom Controls* documentation.

18

Using the Control Editor

▪ Invoking the Control Editor	134
▪ Creating an Editor Extension	136
▪ Adding a Control to an Editor Extension	138
▪ Adding a Data Type to an Editor Extension	139
▪ Deleting a Control or Data Type	141
▪ Saving an Editor Extension	141
▪ Opening an Editor Extension	142
▪ Invoking Help for the Control Editor	142

Invoking the Control Editor

When you invoke the Control Editor, a dialog appears which shows a list of all editor extensions (that is, the XML files in the directory `<your-webapplication>/cis/config` which have the prefix "edit-or_"). A single editor extension may contain multiple controls. Application Designer comes along with a set of predefined editor extensions.



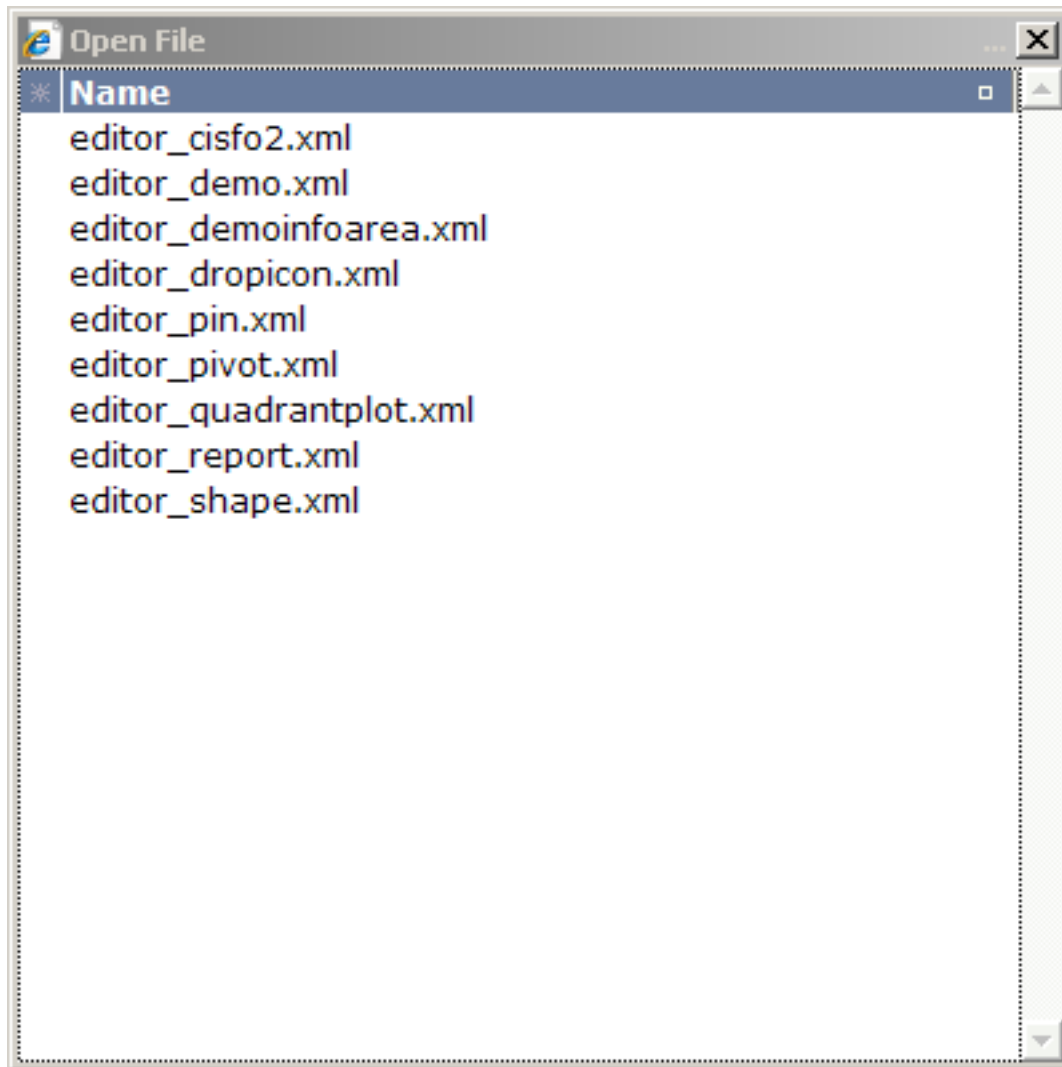
Important: Never touch an editor extension file you do not own. Develop your controls within your own extension.

If you start the Control Editor for the first time, cancel the dialog and create a new editor extension. See [Creating an Editor Extension](#).

▶ To invoke the Control Editor

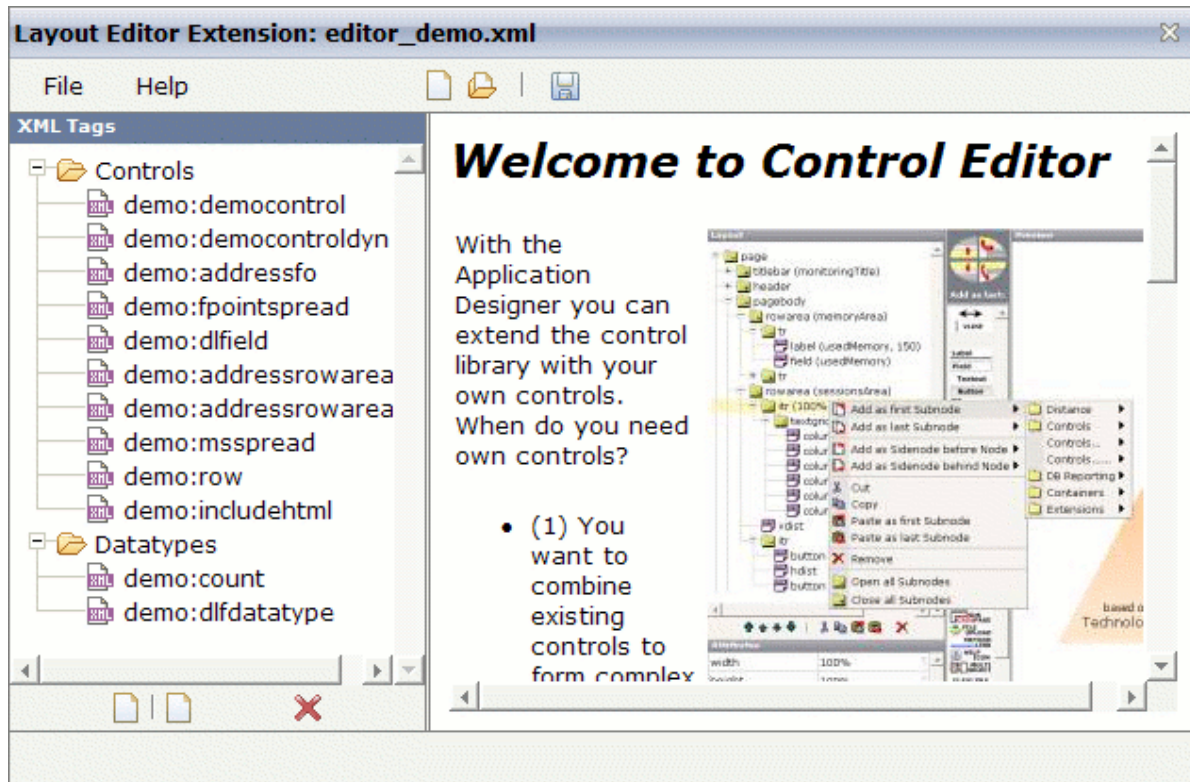
- 1 In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Control Editor**.

The following dialog appears, listing all available editor extensions.



- 2 Choose the editor extension that you want to open.

The contents of the editor extension are loaded into the Control Editor. Example:



You can now edit your editor extension as described in the remainder of this section.

 **Note:** You can also open an editor extension as described in [Opening an Editor Extension](#).

Creating an Editor Extension

When you create a new editor extension, it is stored in the directory `<your-webapplication>/cis/config/`.

► To create an editor extension

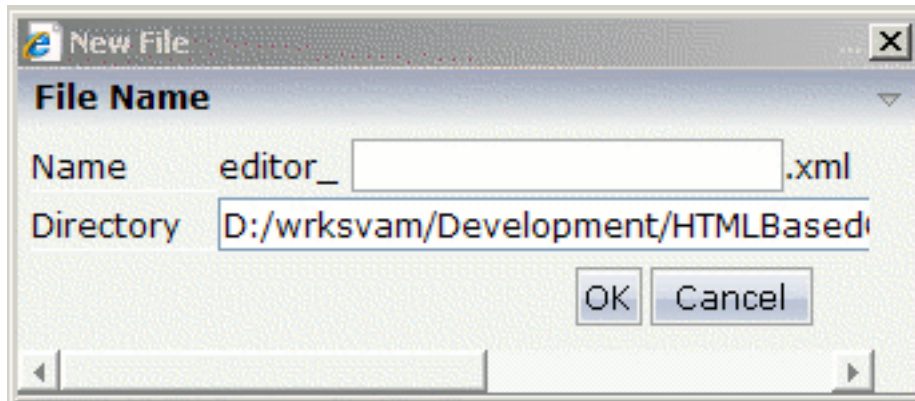
1 From the **File** menu of the Control Editor, choose **New**.

Or:

Choose the following button from the toolbar of the Control Editor.



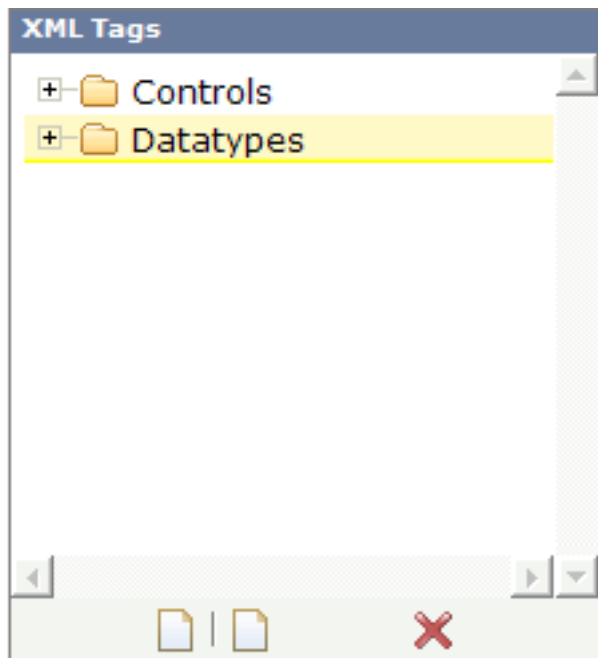
The following dialog appears.



- 2
- 3 Enter the name of your editor extension (for example, "mycontrols").
- 4 Choose the **OK** button.

The name of your new editor extension is composed of the prefix "editor_" and the name you have specified (for example, *editor_mycontrols.xml*). The extension is always *xml*.

The following empty nodes are now shown.



You can now add **controls** and **data types** as described below.

Adding a Control to an Editor Extension

A control consists of the control's definition (attributes and positioning) and its corresponding tag handler (Java class).

When defining macro controls, you do not have to write your own tag handler class. You can define the inner structure directly within XML (XML/protocol extension).

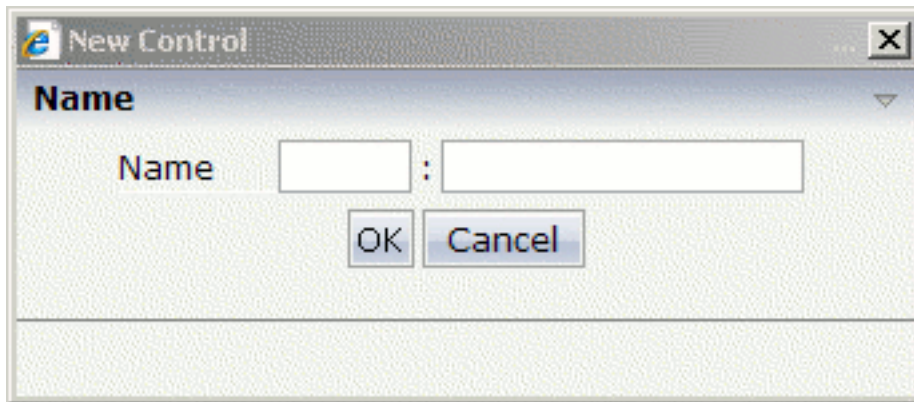
When defining completely new controls which implement their own HTML/JavaScript, you must also implement a corresponding tag handler in order to generate your own HTML/JavaScript code.

▶ To add a control

- 1 Choose the first button which is shown below the tree of XML tags (when you move the mouse over this button, the tooltip "New Control" appears):



The following dialog appears.



- 2 Enter a name for the new control.

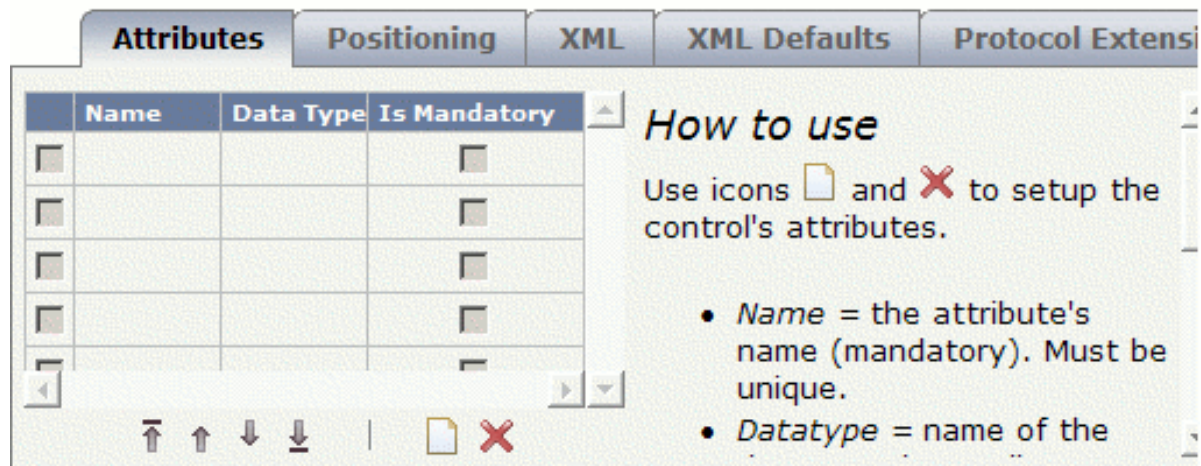
The name of a control must be unique within your library. Therefore, you have to prefix the control name with the name of the library: "`<library-name><control-name>`". See also *Library Concept* in the *Custom Controls* documentation.



- 3 Choose the **OK** button.

The new control is inserted in the **Controls** node. If the **Controls** node already contains controls, each new control is automatically inserted at the bottom of the list.

- 4 Select the new control in the tree.

The following information is shown.



Using the buttons  and  you can hide and show the help information in this dialog.

- 5 Specify all required information on the different tabs. See [Defining a Control](#) for detailed information on these tabs.

See also [Examples](#).

Adding a Data Type to an Editor Extension

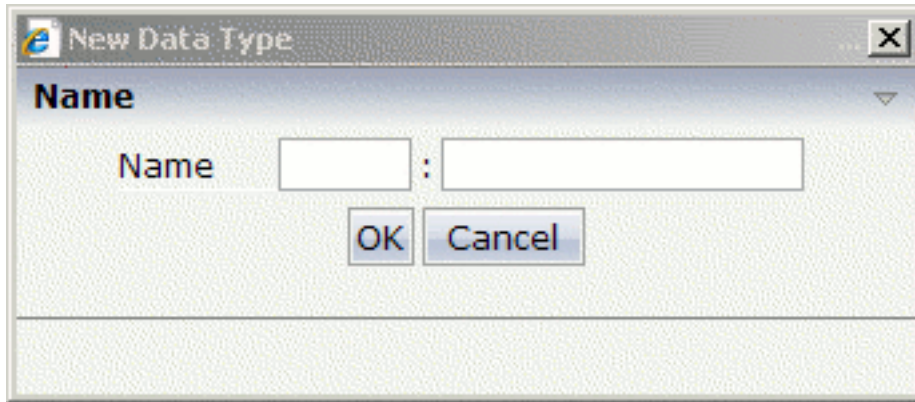
A data type defines a list of valid values. The properties of your controls can have a data type; this is optional. Application Designer provides a set of predefined data types (such as `align` or `boolean`) that you can use within your control's definition. They are defined in the editor configuration file `editor.xml`. If there is no appropriate data type for your purpose, you can create your own data type.

▶ To add a data type

- 1 Choose the second button which is shown below the tree of XML tags (when you move the mouse over this button, the tooltip "New Datatype" appears):



The following dialog appears.



- 2 Enter a name for the new data type.

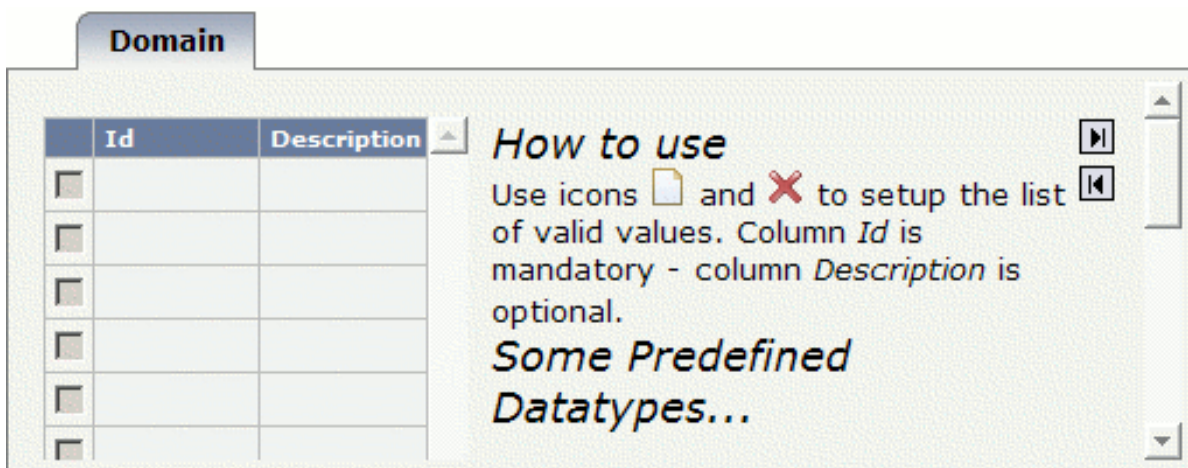
The name of a data type must be unique within your library. Therefore, you have to prefix the data type name with the name of the library: "*<library-name>:<data-type-name>*".



- 3 Choose the **OK** button.

The new data type is inserted in the **Datatypes** node. If the **Datatypes** node already contains data types, each new data type is automatically inserted at the bottom of the list.



- 4 Select the new data type in the tree.


The following information is shown on the right side of the screen.



Using the buttons  and  you can hide and show the help information in this dialog.

The following buttons are provided on this tab:

Button	Description
	Adds an empty line in which you can specify a data type.
	Deletes the selected data type(s).

- 5 Use the  button to add one or more empty lines and specify the following:

Id

The name of the data type.

Description

Optional. A short description of your data type.

See also [Examples](#).

Deleting a Control or Data Type

You can delete any controls or data types that are shown in the tree of XML tags.

▶ To delete a control or data type

- 1 In the tree of XML tags, select the control or data type that you want to delete.
- 2 Choose the following button which is shown below the tree of XML tags:



You are not asked to confirm the deletion.

Saving an Editor Extension

When you save an editor extension, all of your changes in the Control Editor are saved. This includes all controls and data types that you have added or changed.

▶ To save an editor extension

- From the **File** menu of the Control Editor, choose **Save**.

Or:

Choose the following button from the toolbar of the Control Editor.



The status bar of the Control Editor shows the name of the file (including the path) to which the information has been written.

Opening an Editor Extension

You can open any editor extension that is stored in the directory *<your-webapplication>/cis/config/*.

▶ To open an editor extension

- 1 From the **File** menu of the Control Editor, choose **Open**.

Or:

Choose the following button from the toolbar of the Control Editor.



Note: When your latest changes in the Control Editor have not yet been saved, you are asked whether you want to save them.

The **Open File** dialog appears.

- 2 Choose the editor extension that you want to open.

The contents of the editor extension are loaded into the Control Editor.

Invoking Help for the Control Editor

You can invoke the online documentation for the Control Editor.

▶ To invoke help for the Control Editor

- From the **Help** menu of the Control Editor, choose **Documentation**.

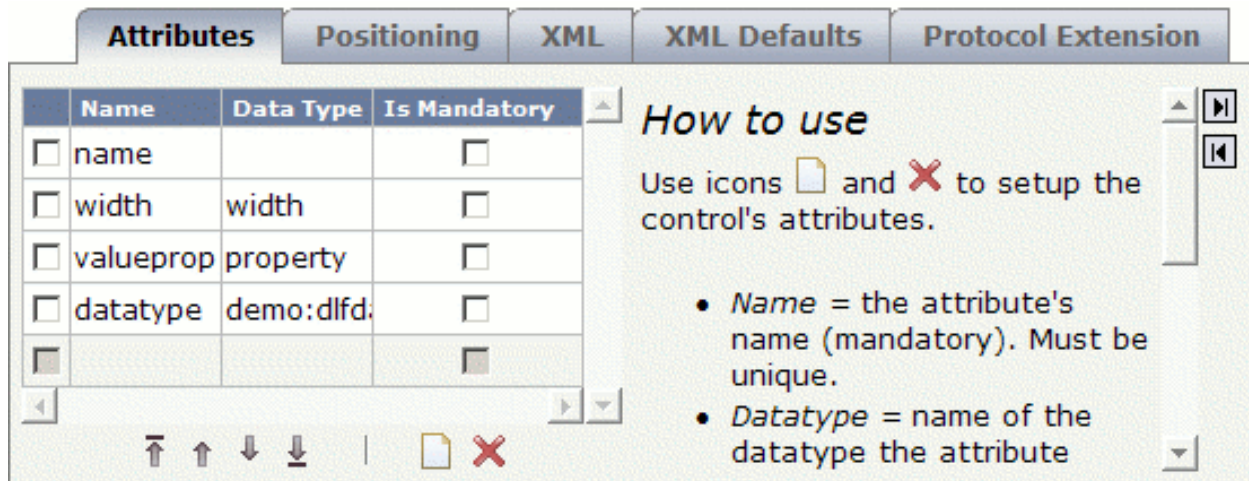
19

Defining a Control

▪ Attributes	144
▪ Positioning	145
▪ XML	147
▪ XML Defaults	148
▪ Protocol Extension	149

Attributes

The **Attributes** tab is used to specify the list of attributes.



The following buttons are provided below the list of attributes:

Button	Description
	Moves the selected attribute(s) up to first position in the list.
	Moves the selected attribute(s) up to the previous position in the list.
	Moves the selected attribute(s) down to the next position in the list.
	Moves the selected attribute(s) down to last position in the list.
	Adds an empty line in which you can specify a property.
	Deletes the selected attribute(s).

To define a property, use the button to add an empty line and specify the following information in this line:

Name

The name of a property. The name must be unique within this attribute list.

Data Type

Optional. The name of a data type to which the attribute refers. The data type can be an Application Designer data type (see the data type definitions in the file *editor.xml*) or a user-defined data type.

Is Mandatory

Optional. When this check box is selected, a value for the corresponding attribute must be set.

In the case of macro controls, the generation protocol will show an error message if input is missing.

Positioning

The **Positioning** tab is used to specify the following:

- the section of the controls palette which is to contain the control, and
- the containers in which the user will be able to insert the control, and
- the subcontrols which the user can insert into the control.

The screenshot shows the **Positioning** tab selected among other tabs: **Attributes**, **XML**, **XML Defaults**, and **Protocol Extension**.

Control Palette
Item Name e.g. Container, Controls

Embedding Containers



Control Name (e.g. itr, tr)
<input type="checkbox"/> itr
<input type="checkbox"/> tr
<input type="checkbox"/>
<input type="checkbox"/>

Sub Controls

Control Name (e.g. icon)
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>



How to use

(1) Embedding Containers
In order to integrate your control into the control hierarchy within Application Designer Layout Editor enter the list of embedding container control here.

Example: You want to be able to insert your control into Application Designer's row container (*itr* and *tr*). For that create a first line (use icon ) and input *itr* - and a second line and input *tr*. Use icon  to remove lines from the list.

- *Control Name* = name of the container control (mandatory). In case of

The following buttons are provided on this tab:

Button	Description
	Adds an empty line in the corresponding list.
	Deletes the selected control(s) from the corresponding list.

Specify the following information:


Controls Palette

The section of the controls palette in which your new controls are to be included. You can either specify an existing section such as **Controls**, or you can specify a new section.

For example, when you specify the name "MyControls", you have to choose the **MyControls** button in the controls palette to display your controls.

Embedding Containers

This list determines the availability in the Layout Painter.

Use the  button to add one or more empty lines and specify the names of the containers (such as ITR or TR) in which your new control can be inserted as a subnode. If specify your own containers, do not forget to use the library prefix.

In the Layout Painter, the new control can then be selected from:

- **Context Menu**

The new control will be offered for selection in the Layout Painter when you invoke the context menu for an embedding container control.

For example, when you define a new control with the name "test:mycontrol" and define the ITR control as an embedding container, this control will be available as follows:

- **Add as first Subnode > Extensions > test:mycontrol**
- **Add as last Subnode > Extensions > test:mycontrol**

- **Controls Palette**

The new control will be available in the controls palette of the Layout Painter. It is shown when you open the corresponding section of the controls palette. For example, when you have defined that control is to appear in the **MyControls** section, you have to choose the **MyControls** button.

The Layout Painter will use a default image in the controls palette. However, you can assign your own image; in this case, you have to observe the following rules:

- The image must be a GIF file.
- The name of the image must have the following structure:

`ctrl<LIBRARY-NAME>_<CONTROL-NAME>.gif`


For example:

ctrlTEST_MYCONTROL.gif

- The image must be stored in the directory `<your-webapplication>/cis/config/controlimages`.
- The preferred width is 16 pixels and the preferred height is 16 pixels.

Sub Controls

Your control can be a container itself (for example, if you want to be able to insert Application Designer's ICON control into your control). In this case, you enter the list of the subcontrols here.

Use the  button to add one or more empty lines and specify the names of the subcontrols (such as ICON). If you specify your own controls, do not forget to use the library prefix.

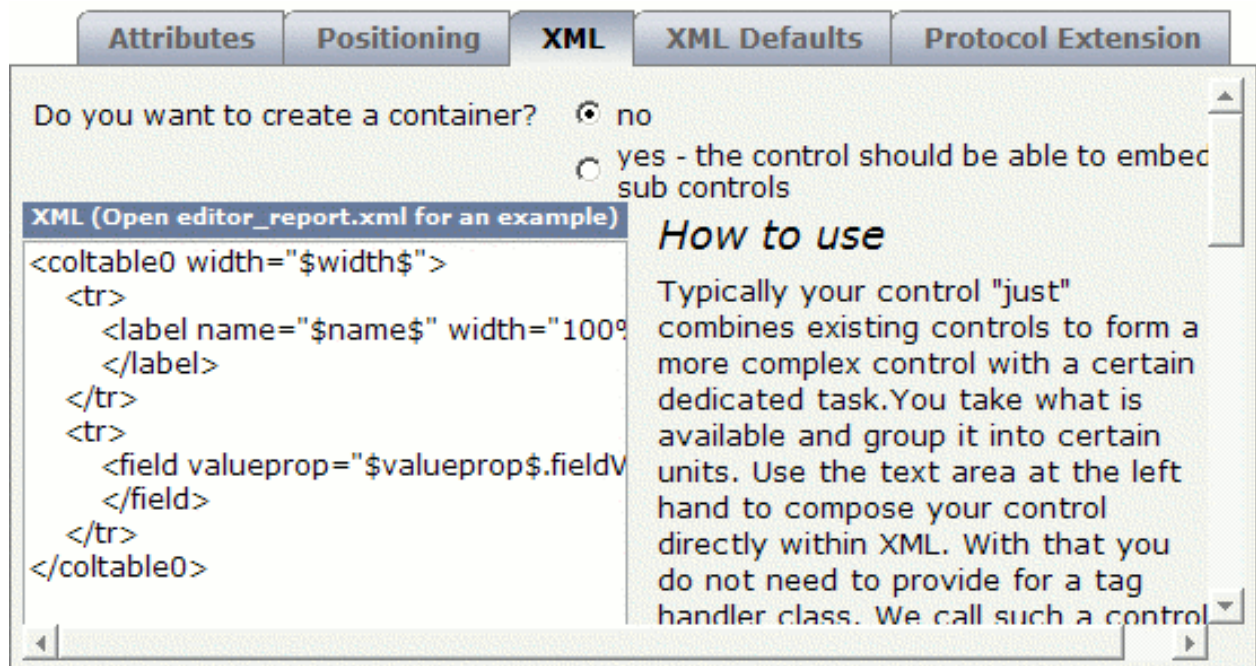
XML

The **XML** tab is used to specify a macro control.

Typically, your control combines existing controls to form a more complex control with a certain dedicated task. You take what is available and group it into certain units.

With a macro control, you compose your control directly in XML and therefore do not need to provide for a tag handler class.

See also [Defining a Macro Control](#) in the *Examples* section.



Specify the following information:

Do you want to create a container?

Select one of the following option buttons:

- **no**
When this button is selected, your control cannot embed other controls. BUTTON is an example of such a control.
- **yes**
When this button is selected, your control is a container control which can embed other controls. ITR and ROWAREA are examples of container controls.

XML

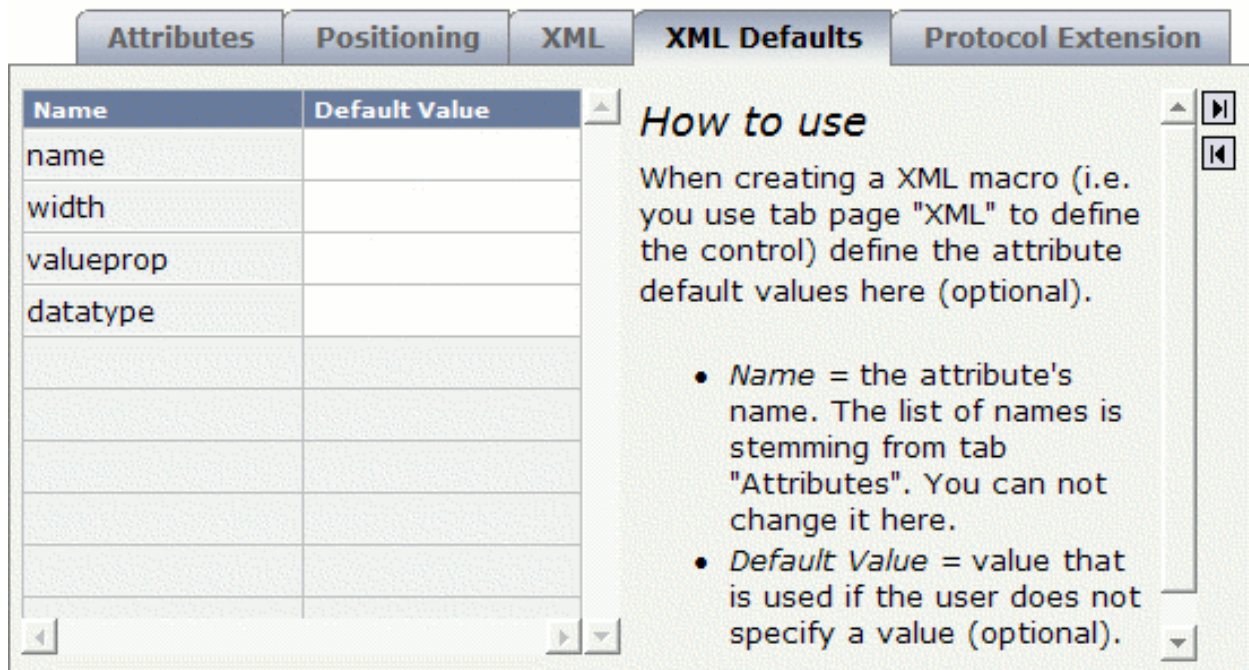
Enter your macro in this text box.

XML Defaults

Only applies to a macro control.

When you have entered a macro control on the **XML** tab, you can use the **XML Defaults** tab to define the default values for the attributes (optional).

All attributes that are currently defined on the **Attributes** tab are automatically provided on the **XML Defaults** tab.



Specify the following information:

Default Value

The default value will be used if the user does not specify a value.

Protocol Extension

Use the **Protocol Extension** tab only if you develop a macro control and do not provide a tag handler class. In this case, you can use this tab to append the referred properties and methods to the generation protocol. If you provide a tag handler class, you implement the addition of properties and methods to the generation protocol in the tag handler class instead.

See also [XML Definition for a Macro Control with a Server-Side Representative](#) in the *Examples* section.

Attributes **Positioning** **XML** **XML Defaults** **Protocol Extension**

Properties

Name	Data Type	Preset	Show in
<input type="checkbox"/> \$valueprops:DLFIELDInfo			<input checked="" type="checkbox"/>
<input type="checkbox"/> \$valueprops:String			<input type="checkbox"/>
<input type="checkbox"/> \$valueprops:boolean			<input type="checkbox"/>

Methods

Method Name	Show in
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>

Javascript Libraries

Source (e.g. "../yourproject/javascript/lib.js")
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>



How to use

Use tab *Protocol Extension* only if you develop a XML Macro. Publish the referred properties and methods here. If you provide for a tag handler it is the job of that class to append additional properties and methods to the generation protocol.

(1) Properties


- *Name* = name of the adapter property that is used within your control (mandatory).
- *Datatype* = the property's datatype (optional). This information is used within the Code Assistant of the Layout Editor to generate appropriate property coding.
- *Preset Value* = value that set into your adapter when the page is loading (optional).

The following buttons are provided on this tab:

Button	Description
	Adds an empty line in the corresponding list.
	Deletes the selected line(s) from the corresponding list.

Specify the following information:

Properties

Use the  button to add one or more empty lines and specify the following:

Name

The name of the adapter property that is used in your control.

Data Type

Optional. The data type of the property. This information is used in the Code Assistant (which is part of the Layout Painter) to generate appropriate property coding.



Note: This has no meaning for custom controls in NATPAGE layouts.

Preset Value

Optional. The value that is set in your adapter when the page is loading.

Show in Code Assistant


When this check box is selected, the property appears in the Code Assistant. By default, this check box is selected.

You should only select this check box for the properties that must be provided by the adapter.



Note: This has no meaning for custom controls in NATPAGE layouts.

Methods

Use the  button to add one or more empty lines and specify the following:

Method Name

The name of the adapter method.

Show in Code Assistant


When this check box is selected, the method appears in the Code Assistant. By default, this check box is selected.

You should only select this check box for the methods that must be provided by the adapter.



Note: This has no meaning for custom controls in NATPAGE layouts.

JavaScript Libraries

Use the  button to add one or more empty lines and specify the following:

Source

The URL that points to your JavaScript library.

20 Examples

- About the Examples 154
- Defining a Control with a Corresponding Tag Handler 154
- Defining a Macro Control 157
- Additional Information 166

About the Examples

These examples assume that you expect the label of an input field not at the left of the field but above the field (two separate rows). Furthermore, you want to have several such fields within a single row (inline control).

For these examples, you will proceed as follows:

1. You create your own editor extension with the name *editor_test.xml*.



Important: Do not use the Application Designer editor extensions for your own controls. Place your own controls within your own extension file. Thus, there will be no conflicts when upgrading your installation to a newer Application Designer build.

2. You build a control named DLFIELD (double line field) which will be available in the library "test". Therefore, you have to name your control "test:dlfield". See also *Library Concept* in the *Custom Controls* documentation.
3. You define a macro control which determines that the control consists of two rows, one for the label and another for the field. The rows themselves are placed into a column container.

The composition of controls looks roughly like this:

```
COLTABLEO
  TR
    LABEL
  TR
    FIELD
```

Defining a Control with a Corresponding Tag Handler

The definition of a control with a corresponding tag handler is quite simple. It consists of

- the name of the control,
- the list of attributes, and
- the list of embedding container controls.


You will now define a control with the following properties: *name* (label name), *width* (width of column container) and *valueprop* (name of the adapter property to which the field is bound). The control will be available within the row containers TR and ITR.



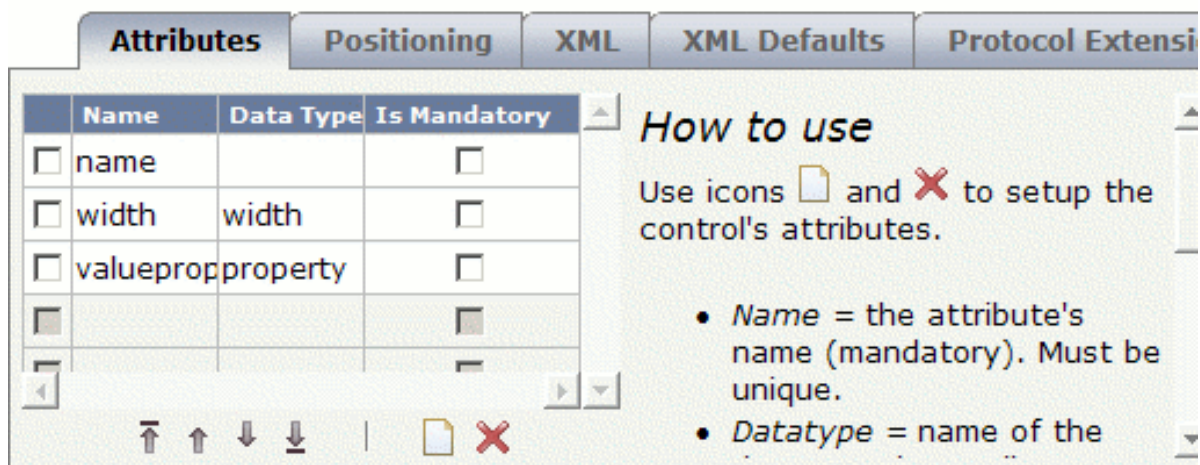
Note: For further information on the tag handler, see *Control Concept* in the *Custom Controls* documentation.

▶ **To define a new control**

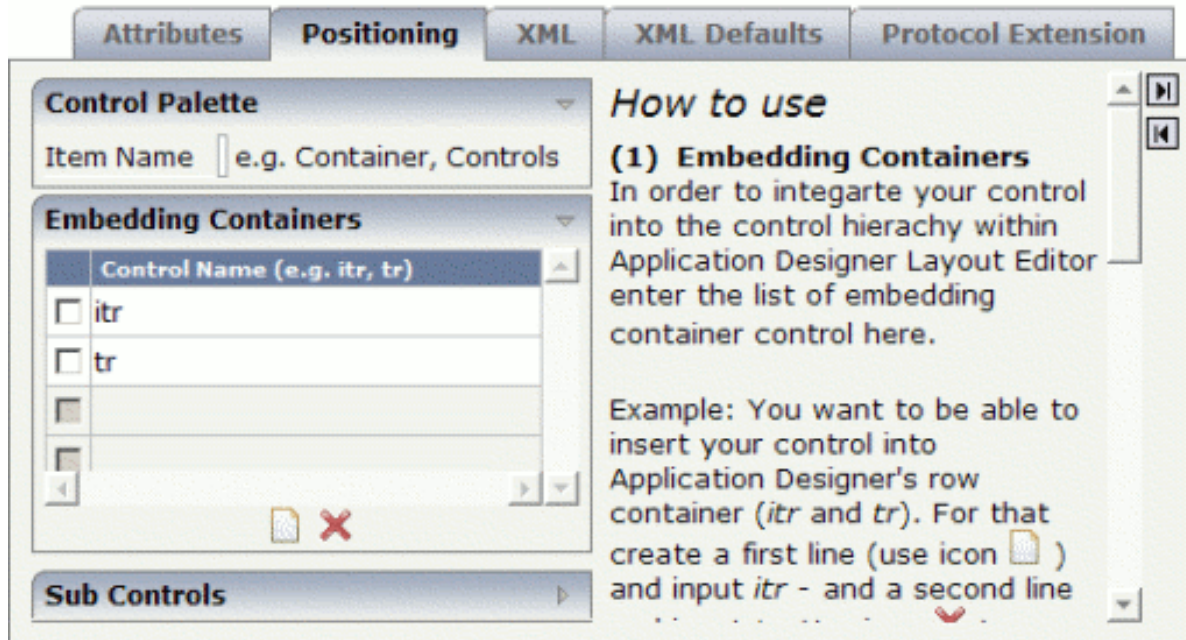
- 1 Invoke the Control Editor.
- 2 **Create your own editor extension** with the name *editor_test.xml*
- 3 **Add a new control** with the name "test:dlfield".

 **Note:** The editor extension *editor_demo.xml* contains a control with the name "demo:dlfield". When you have completed the example, your new "test:dlfield" extension should be similar to "demo:dlfield".

- 4 Select your new control in the tree.
- 5 To create the list of attributes, specify the following information on the **Attributes** tab:



- 6 Specify the following information on the **Positioning** tab:



This determines that your control can be selected from the **MyControls** section of the controls palette and that it can be inserted into the row containers TR and ITR.

- 7 Save your changes.
- 8 Have a look at the generated XML file (*editor_test.xml*). It should look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
Dynamic extension of editor.xml file.
-->

<controllibrary>
  <editor>

  <!--
  *****
  * DATATYPES
  *****
  -->

  <!--
  *****
  * TAGS
  *****
  -->

  <!-- TEST:DLFIELD -->
  <tag name="test:dfield">
```

```

    <attribute name="name"/>
    <attribute name="width" datatype="width"/>
    <attribute name="valueprop" datatype="property"/>
    <taginstance>
    </taginstance>
    <protocolitem>
    </protocolitem>
  </tag>
  <tagsubnodeextension control="itr" newsubnode="test:dfield"/>
  <tagsubnodeextension control="tr" newsubnode="test:dfield"/>

  <taggroupsubnodeextension group="MyControls" newsubnode="test:dfield"/>

</editor>
</controllibrary>

```

Defining a Macro Control

You can define a macro control as XML without any additional coding. The basic specification for a macro control defined as XML is the same as the specification for a control to which also a tag handler is applied. It consists of:

- the name of the control,
- the list of attributes, and
- the list of embedding container controls.

In addition, you define the following:

- the XML for the control,
- additional properties and methods (optional), and
- additional JavaScript libraries (optional).

With the definition of a macro control, you describe how the control is composed out of other controls. An attribute of the macro control can be referenced by enclosing the attribute name in "\$" characters (for example, "\$myattribute\$").

With a simple macro control, no further information is required.

Additional data (additional properties, methods, libraries) is only required if you want to bind the control to a so-called “server-side representative”. The control's properties and methods are bound to a dedicated Java class. Within this class, you typically encapsulate certain tasks/functionality used by the control.

The following topics are covered below:

- [XML Definition for a Simple Macro Control](#)
- [XML Definition for a Macro Control with a Server-Side Representative](#)

XML Definition for a Simple Macro Control

You will now continue with the "test:dlfield" control from the previous example.

You will add a macro control to the "test:dlfield" control which defines the following:

- The control consists of a row for the label and a second row for the field.
- The rows themselves are placed into a column container.
- The property `width` is delegated to the column container. The label and field have a width of 100%.
- The property `name` is delegated to the corresponding property of the label.
- The property `valueprop` is delegated to the corresponding property of the field.

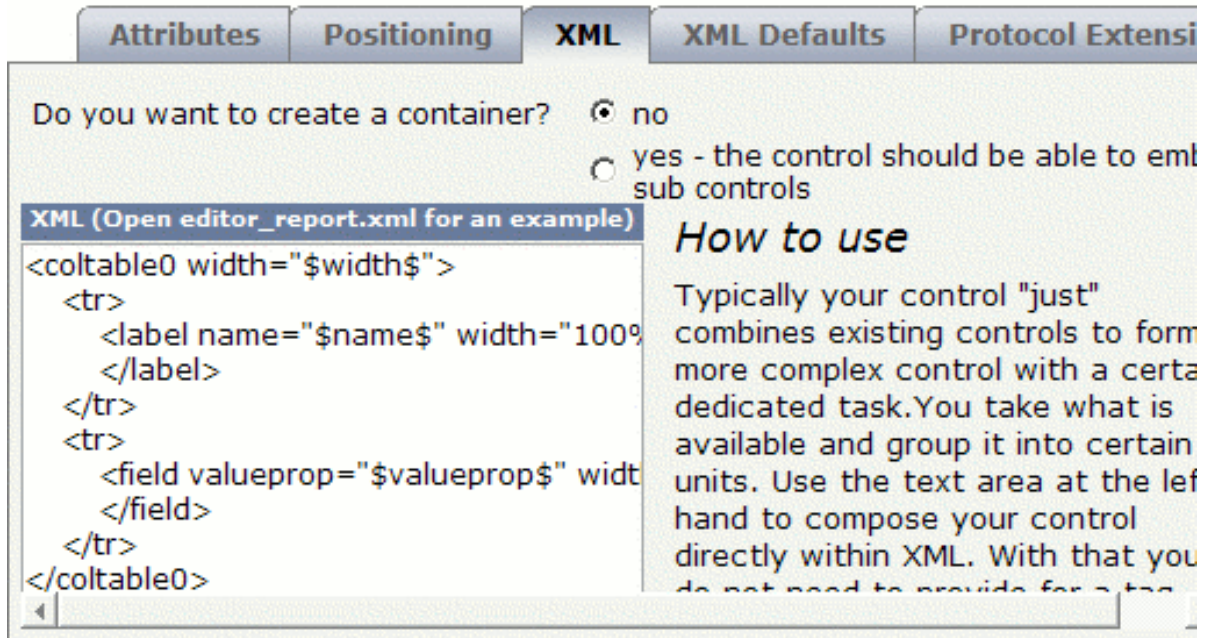
The XML for the macro control looks like this:

```
<coltable0 width="$width$">
  <tr>
    <label name="$name$" width="100%" asplaintext="true">
    </label>
  </tr>
  <tr>
    <field valueprop="$valueprop$" width="100%">
    </field>
  </tr>
</coltable0>
```

▶ To define the XML of the macro control

- 1 Specify the above XML on the **XML** tab of the "test:dlfield" control.

The tab should now look as follows:



- 2 Save your changes.
- 3 Have a look at the generated XML file (*editor_test.xml*). It should now look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Dynamic extension of editor.xml file.
-->
<controllibrary>
  <editor>

  <!--
  *****
  * DATATYPES
  *****
  -->

  <!--
  *****
  * TAGS
  *****
  -->

  <!-- TEST:DLFIELD -->
  <tag name="test:dfield">
    <attribute name="name"/>
    <attribute name="width" datatype="width"/>
```

```

<attribute name="valueprop" datatype="property"/>
<taginstance>
  <coltable0 width="$width$" >
    <tr>
      <label name="$name$" width="100%" asplaintext="true">
      </label>
    </tr>
    <tr>
      <field valueprop="$valueprop$" width="100%">
      </field>
    </tr>
  </coltable0>
</taginstance>
<protocolitem>
</protocolitem>
</tag>
<tagsubnodeextension control="itr" newsubnode="test:dfield"/>
<tagsubnodeextension control="tr" newsubnode="test:dfield"/>

<taggroupsubnodeextension group="MyControls" newsubnode="test:dfield"/>

</editor>
</controllibrary>

```

The control is now ready for use. The Layout Painter will offer the control for the containers ITR and TR.

XML Definition for a Macro Control with a Server-Side Representative

This example demonstrates how to build a macro control that refers to a server-side representative.

The server-side representative class `DLFIELDInfo` described below is specific for custom controls used within PAGE layouts. In addition, the Code Assistant support described below is only available in PAGE layouts.

For NATPAGE layouts, you can also implement server-side representative classes. These are the so-called "binding classes" (see also *Binding Concept* in the *Custom Controls* documentation).

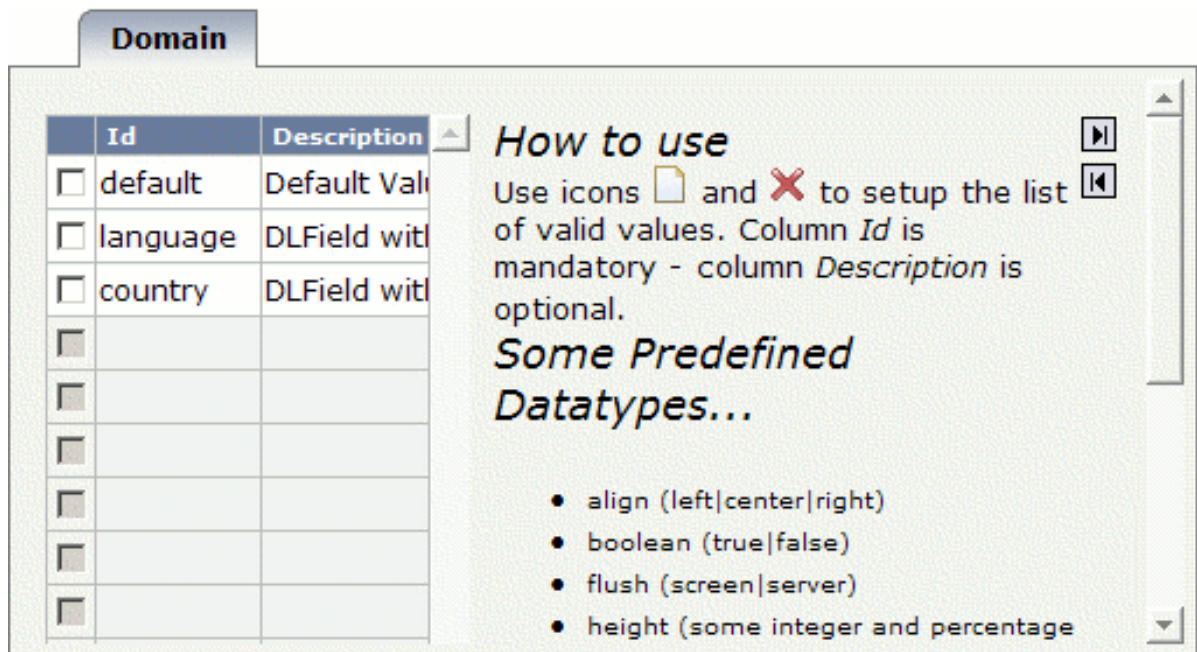
This example demonstrates how to build a macro control that refers to a server-side representative. You will now extend the "test:dfield" control from the previous example as follows:

- If the field is used to enter a country or language, it provides a value help.
- For reuse, the computation of valid values is encapsulated within a server-side representative (Java class `DLFIELDInfo`).
- The field (i.e. the properties `valueprop` and `popupprop`) is bound to properties of class `DLFIELDInfo`.

Thus, the page adapter just has to provide for a `DLFIELDInfo` object; it does not have to compute the value help by itself.

► To define a macro control with a server-side representative

- 1 **Add a new data type** with the name "test:dlfdatatype".
- 2 Select the new data type in the tree.
- 3 Specify the following values on the **Domain** tab:



- 4 Select the control "test:dlfield" in the tree.
- 5 On the **Attributes** tab, add the attribute datatype and define "test:dlfdatatype" as the data type.

Name	Data Type	Is Mandatory
<input type="checkbox"/> name		<input type="checkbox"/>
<input type="checkbox"/> width	width	<input type="checkbox"/>
<input type="checkbox"/> valueprop	property	<input type="checkbox"/>
<input type="checkbox"/> datatype	demo:dlfc	<input type="checkbox"/>
<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>		<input type="checkbox"/>

How to use
Use icons and to setup the control's attributes.

- *Name* = the attribute's name (mandatory). Must be unique.
- *Datatype* = name of the datatype the attribute refers to (optional).
Purpose: provides for a list of valid values within the Application Designer Layout

Until now, you have used the attribute `valueprop` to bind the field's input value to an adapter property.

Using the server-side representative concept, the adapter now provides a property (referenced by the property `valueprop`) that returns an instance of such a representative (`DLFIELDInfo`). The field value and the value help are handled within that class. The following shows the coding of the class `DLFIELDInfo`:

```
package com.softwareag.cis.test35;

import com.softwareag.cis.server.util.ValidValueLine;
import java.util.*;
import com.softwareag.cis.server.*;
import com.softwareag.cis.server.util.*;
import com.softwareag.cis.util.*;

public class DLFIELDInfo
{
    private final static String ML_APP = "release35";
    private Adapter m_adapter;

    //Constructor

    public DLFIELDInfo(Adapter adapter)
    {
        m_adapter = adapter;
    }

    // -----
    // properties
```

```
// -----  
  
// property >fieldValue<  
String m_fieldValue;  
public String getFieldValue() { return m_fieldValue; }  
public void setFieldValue(String value) { m_fieldValue = value; }  
  
// property >hasPopupHelp<  
public boolean getHasPopupHelp()  
{  
    return m_fieldDatatype != null &&  
        m_fieldDatatype.length() != 0;  
}  
  
// property >fieldDatatype<  
String m_fieldDatatype;  
public String getFieldDatatype() { return m_fieldDatatype; }  
public void setFieldDatatype(String value) { m_fieldDatatype = value; }  
  
// -----  
// methods  
// -----  
  
/** */  
public ValidValueLine[] findValidValuesForFieldValue()  
{  
    String text1 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_1");  
    String text2 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_2");  
    String text3 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_3");  
    String text4 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_4");  
    String text5 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_5");  
    String text6 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_6");  
    String text7 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_7");  
    String text8 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_8");  
    String text9 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_9");  
    String text10 = m_adapter.replaceLiteral(ML_APP, "dlfi_code_10");  
  
    if (m_fieldDatatype != null &&  
        m_fieldDatatype.equalsIgnoreCase(text1))  
    {  
        return new ValidValueLine[]  
        {  
            new ValidValueLine(text2, text3),  
            new ValidValueLine(text4, text5)  
        };  
    }  
    if (m_fieldDatatype != null &&  
        m_fieldDatatype.equalsIgnoreCase(text6))  
    {  
        return new ValidValueLine[]  
        {  
            new ValidValueLine(text2, text7),
```

```

        new ValidValueLine(text8, text9)
    };
}
throw new Error(text10);
}
}

```

6 Select the **XML** tab and change the existing macro as follows:

```

<coltable0 width="$width$" >
  <tr>
    <label name="$name$" width="100%" asplaintext="true">
    </label>
  </tr>
  <tr>
    <field valueprop="$valueprop$.fieldValue" popupmethod="openIdValueCombo" ←
    popupprop="$valueprop$.hasPopupHelp" width="100%">
    </field>
  </tr>
</coltable0>

```

7 Add the used properties to the **Protocol Extension** tab so that it looks as follows.

Name	Data Type	Preset Value	Show in
<input type="checkbox"/> \$valueprop\$	DLFIELDInfo		<input checked="" type="checkbox"/>
<input type="checkbox"/> \$valueprop\$.fieldVal	String		<input type="checkbox"/>
<input type="checkbox"/> \$valueprop\$.hasPop	boolean		<input type="checkbox"/>
<input type="checkbox"/> \$valueprop\$.fieldDat	String	\$datatype\$	<input type="checkbox"/>

generation protocol.

(1) Properties

- *Name* = name of the adapter property that is used within your control (mandatory).
- *Datatype* = the property's datatype (optional). This information is used within the Code Assistant of the Layout Editor to generate appropriate property

The properties `$valueprop$`, `$valueprop$.fieldValue` and `$valueprop$.hasPopupHelp` will be included in the access path of the layout.

You can specify whether a property is to be shown within the Code Assistant of the Layout Painter. Select only the properties that must be provided by the adapter itself (in this example, this is property `$valueprop$`). Do not select the properties that are provided by the control representative. This class is written once and used multiple times within your adapters.

The property `$valueprop$.fieldDatatype` is used to pass the value of the attribute `datatype` from the control to the class `DLFIELDInfo`. The value is set once when the page is loaded. With this, the `DLFIELDInfo` class determines whether pop-up help is available and computes the list of valid values for the pop-up help.

- 8 Save your changes.
- 9 Have a look at the generated XML file (*editor_test.xml*). It should now look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
Dynamic extension of editor.xml file.
-->

<controllibrary>
  <editor>

    <!--
    *****
    * DATATYPES
    *****
    -->

    <!-- TEST:DLFDATATYPE -->
    <datatype name="test:d1fdatatype">
      <value id="default" name="Default value."/>
      <value id="language" name="DLFIELD with language code."/>
      <value id="country" name="DLFIELD with cuntry code."/>
    </datatype>

    <!--
    *****
    * TAGS
    *****
    -->

    <!-- TEST:DLFIELD -->
    <tag name="test:d1field">
      <attribute name="name"/>
      <attribute name="width" datatype="width"/>
      <attribute name="valueprop" datatype="property"/>
      <attribute name="datatype" datatype="test:d1fdatatype"/>
      <taginstance>
        <coltable0 width="$width$">
          <tr>
            <label name="$name$" width="100%" asplaintext="true">
            </label>
          </tr>
          <tr>
            <field valueprop="$valueprop$.fieldValue" popupmethod="openIdValueCombo" ←
popupprop="$valueprop$.hasPopupHelp" width="100%">
            </field>
          </tr>
        </coltable0>
      </taginstance>
```

```
<protocolitem>
  <addproperty name="$valueprop$" datatype="DLFIELDInfo" ↵
useincodegenerator="true"/>
  <addproperty name="$valueprop$.fieldValue" datatype="String"/>
  <addproperty name="$valueprop$.hasPopupHelp" datatype="boolean"/>
  <addproperty name="$valueprop$.fieldDatatype" datatype="String" ↵
presetvalue="$datatype$"/>
</protocolitem>
</tag>
<tagsubnodeextension control="itr" newsubnode="test:dlfield"/>
<tagsubnodeextension control="tr" newsubnode="test:dlfield"/>

<taggroupsubnodeextension group="MyControls" newsubnode="test:dlfield"/>

</editor>
</controllibrary>
```

The control is now ready for use. The Layout Painter will offer the control for the containers ITR and TR.

Additional Information

You can find the sources of the above described DLFIELD example in the demo workplace and in the cisdemos:

- Demo workplace: Open the section **Release 1.3.5** and start the page **Macro Controls > Simple Example**.
- cisdemos:
 - Layout: `<your-webapplication>/cisdemos/xml/35_dlfield.xml`.
 - Adapter: `<your-webapplication>/cisdemos/src/com/softwareag/cis/test35/DLFIELDAdapter.java`.
 - Server-side representative: `<your-webapplication>/cisdemos/src/com/software-ag/cis/test35/DLFIELDInfo.java`.

V

▪ 21 Monitoring	169
▪ 22 Layout Check	173
▪ 23 Performance Tools	177
▪ 24 Developer Documents and License Information	185
▪ 25 Special Tools	189

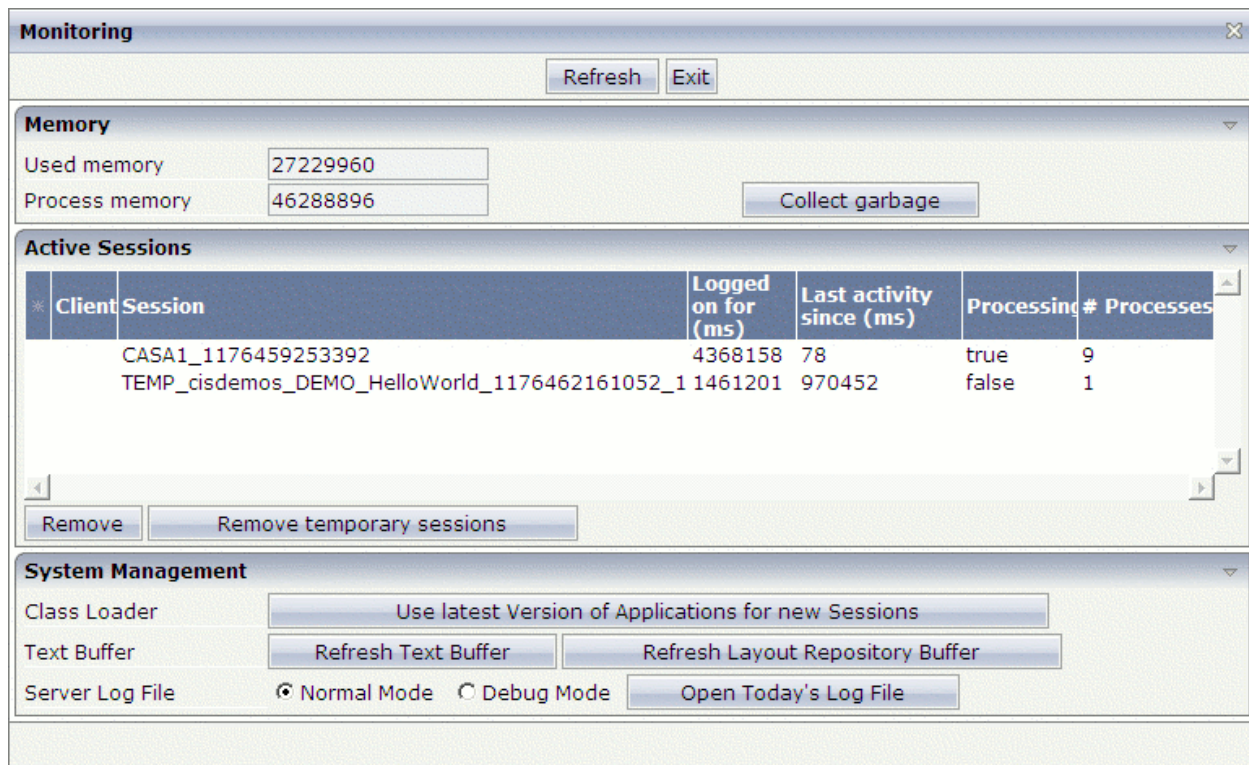
21 Monitoring

▪ Invoking the Monitoring Tool	170
▪ Memory	171
▪ Active Sessions	171
▪ System Management	172

The monitoring tool is used to monitor the sessions that are currently running. Sessions which are no longer required can be deleted and memory can thus be freed. For more information on sessions, see *Details on Session Management* in the *Special Development Topics*.

Invoking the Monitoring Tool

When you invoke the monitoring tool, the following dialog appears.



The **Monitoring** dialog is subdivided into several areas:

- **Memory**
- **Active Sessions**
- **System Management**

Using the **Refresh** button at the very top of the dialog, you can reload the **Monitoring** dialog and thus show the most recent information.

▶ To invoke the monitoring tool

- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Monitoring**.

Or:

Choose the following icon at the top of the development workplace:



Memory

The top of the **Monitoring** dialog provides the following information:

Option	Description
Used memory	This text box indicates the currently used memory.
Process memory	This text box indicates the available memory.
Collect garbage	When you choose this command button, memory which is no longer used on the server is set free.

Active Sessions

The middle of the **Monitoring** dialog provides a list with all sessions which are currently active.

The monitoring process itself is also considered as a session. When no other sessions are active, the monitoring process is the only session that is listed in the **Monitoring** dialog.

The following command buttons are available:

Command Button	Description
Remove	When you choose this command button, the selected sessions are removed from the list of active sessions.
Remove temporary sessions	When you choose this command button, all sessions (except the monitoring session itself) are removed from the list of active sessions.

System Management

The bottom of the **Monitoring** dialog provides the following information:

Option	Description
Class Loader	<p>When you choose the Use latest Version of Applications for new Sessions command button, a new class loader instance is generated. The most recent versions of your applications are then loaded. For example, when an adapter class has been modified, the modified class is loaded and any changes are immediately in effect.</p> <p>See also <i>Class Loader Concepts</i> in the <i>Appendices</i>.</p>
Text Buffer	<p>When you choose the Refresh Text Buffer command button, the most recent versions of your language files are loaded and any changes are immediately in effect.</p> <p>When you choose the Refresh Layout Repository Buffer command button, the most recent versions of your layouts are loaded and any changes are immediately in effect.</p>
Server Log File	<p>The server log file contains a protocol of all interactions with the server. You can select the option button for the required log mode: Normal Mode or Debug Mode. In debug mode, the log contains more detailed information.</p> <p>When you choose the Open Today's Log File command button, a dialog appears showing the content of the log file. See Server Log for further information on this dialog.</p>

22

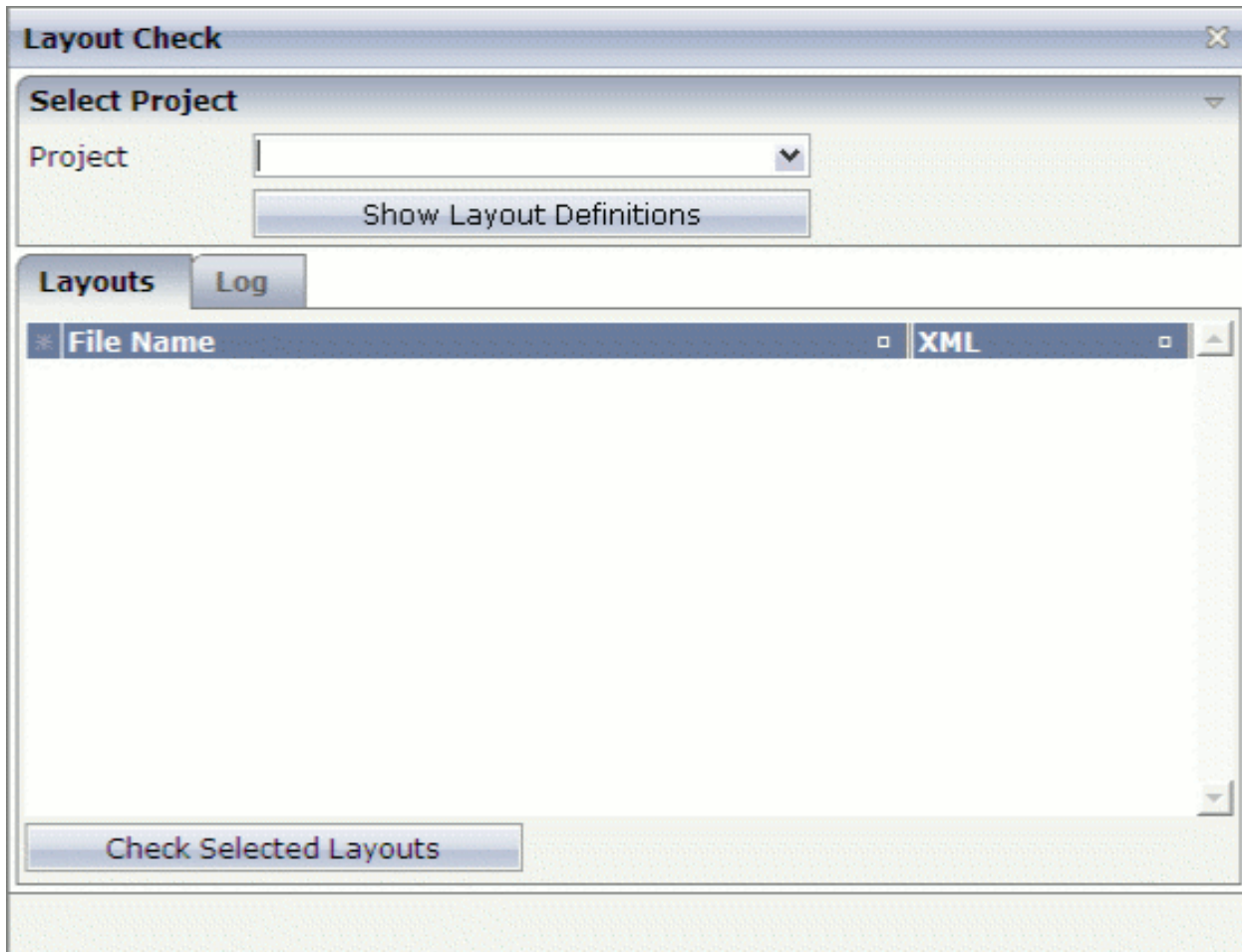
Layout Check

- Invoking the Layout Check Tool 174
- Displaying the Layout Definitions for a Project 175
- Checking Selected Layout Definitions 176

With the layout check tool, you can validate your layout definitions. It lists, for example, any missing properties or invalid definitions.

Invoking the Layout Check Tool

When you invoke the layout check tool, the following dialog appears.



▶ To invoke the layout check tool

- In the **Development Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Layout Check**.

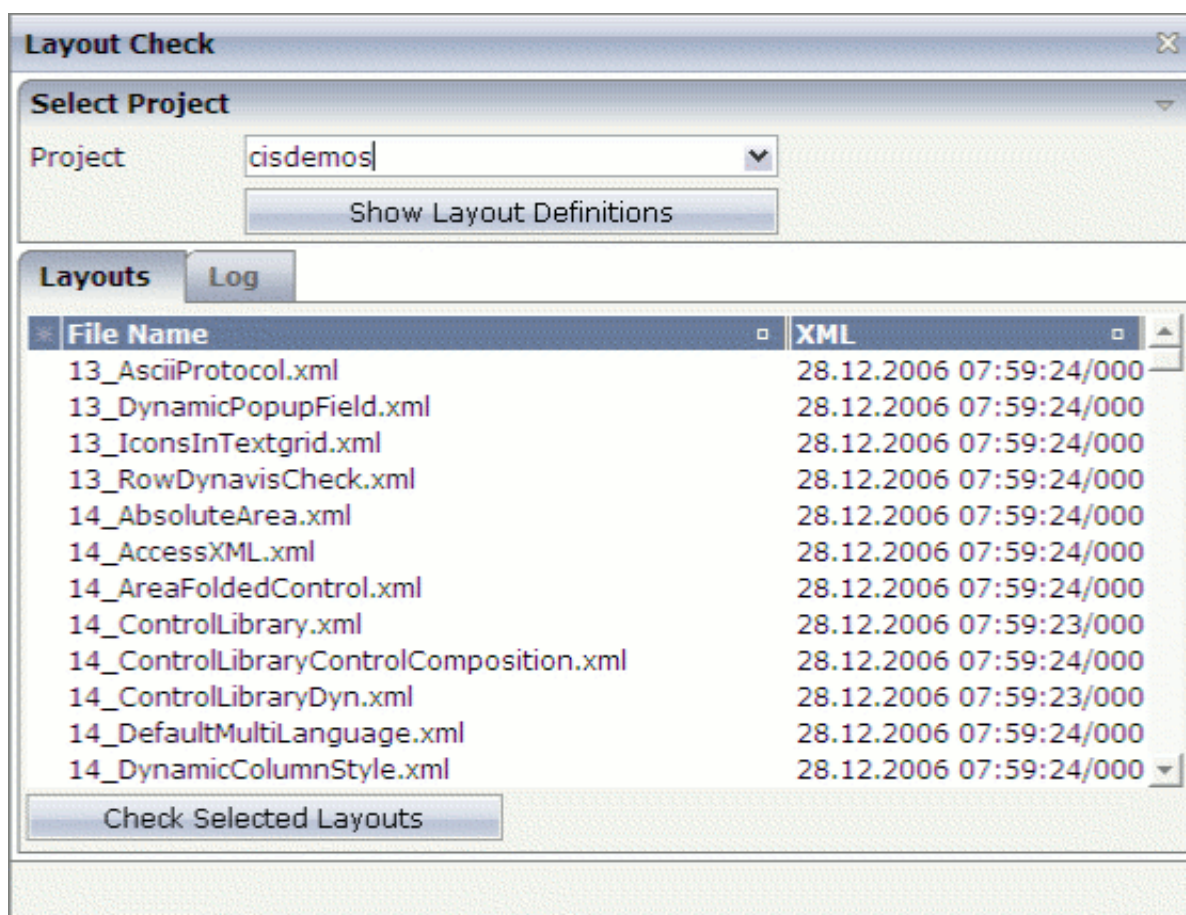
Displaying the Layout Definitions for a Project

In the layout check tool, you first have to specify the project for which you want to check the layouts.

► To display the layout definitions for a project

- From the **Project** drop-down list box, select the required project.

The layout definitions for the selected project are shown. For example:



For each layout definition, the date and time of the last change of the XML file are shown. In the time, the number after the slash indicates a thousandth of a second.



Notes:

1. For each preview, a file with the name *ZZZZZZZZGenerated.xml* is automatically generated.

2. When you have not selected a specific project and choose the **Show Layout Definitions** button, the layout definitions of all projects are shown.

Checking Selected Layout Definitions

You can check several layout definitions at the same time.

▶ To check selected layout definitions

- 1 Select the layout definition(s) that you want to check. See also [Selecting Layout Definitions](#) in the description of the Layout Manager.
- 2 Choose the **Check Selected Layouts** button.

The selected layout definitions are checked.

The row for each checked layout definition which does not contain an error is shown with a green background color.

- 3 Choose the **Log** tab.

For each checked layout definition, a message text is provided which indicates whether the layout definition could be validated successfully or whether it contains an error. In the case of an error, an error description is shown (for example, the name of a missing property or of an invalid subnode).

23 Performance Tools

- Recording a Performance Trace 178
- Executing a Performance Trace 181

The performance tools can be used to test the performance of a web application. First, you record a performance trace (that is, your interaction with the server), and then you execute the recorded performance trace in order to determine the amount of time that is required for processing the server requests.

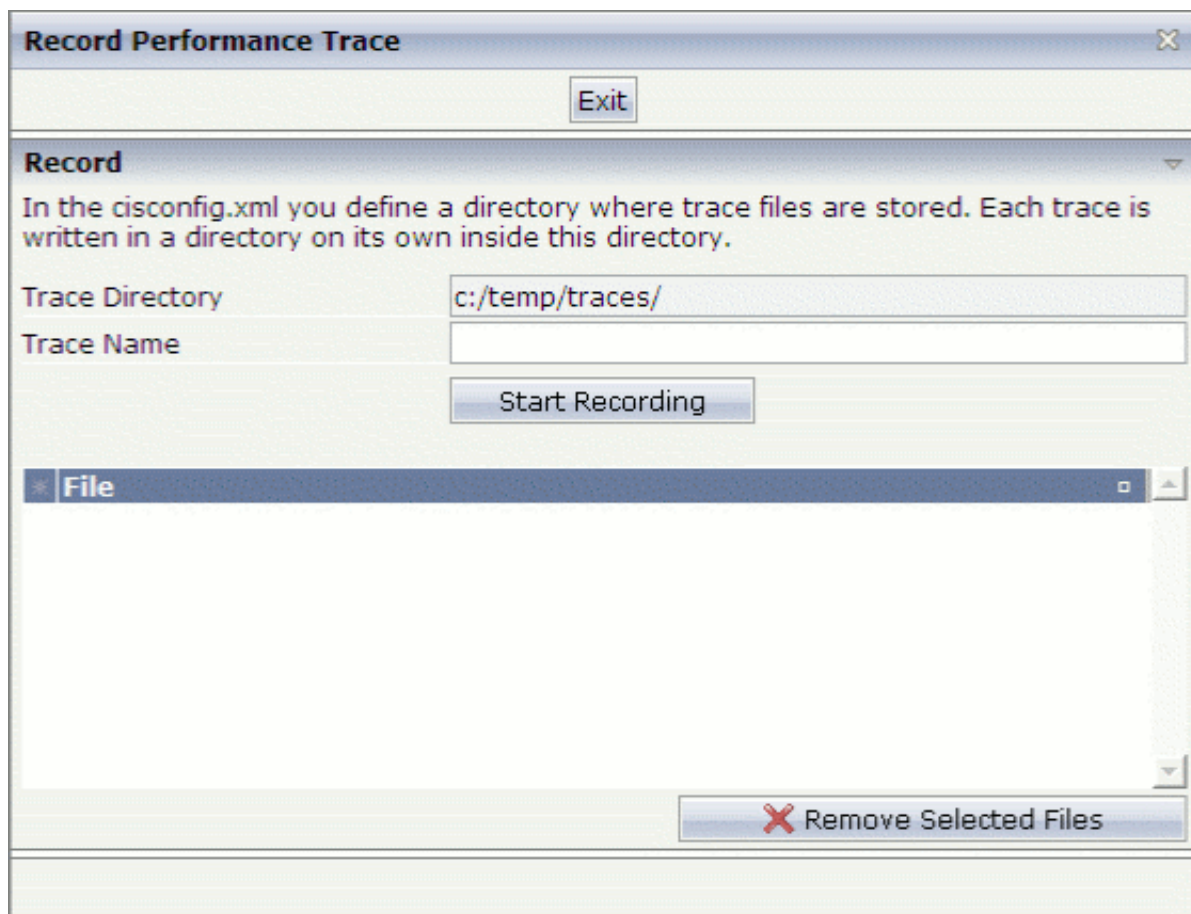
Recording a Performance Trace

Performance traces are recorded using the Start/Stop Trace tool.

▶ To record a performance trace


- 1 In the **Performance Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Start/Stop Trace**.

The following dialog appears.



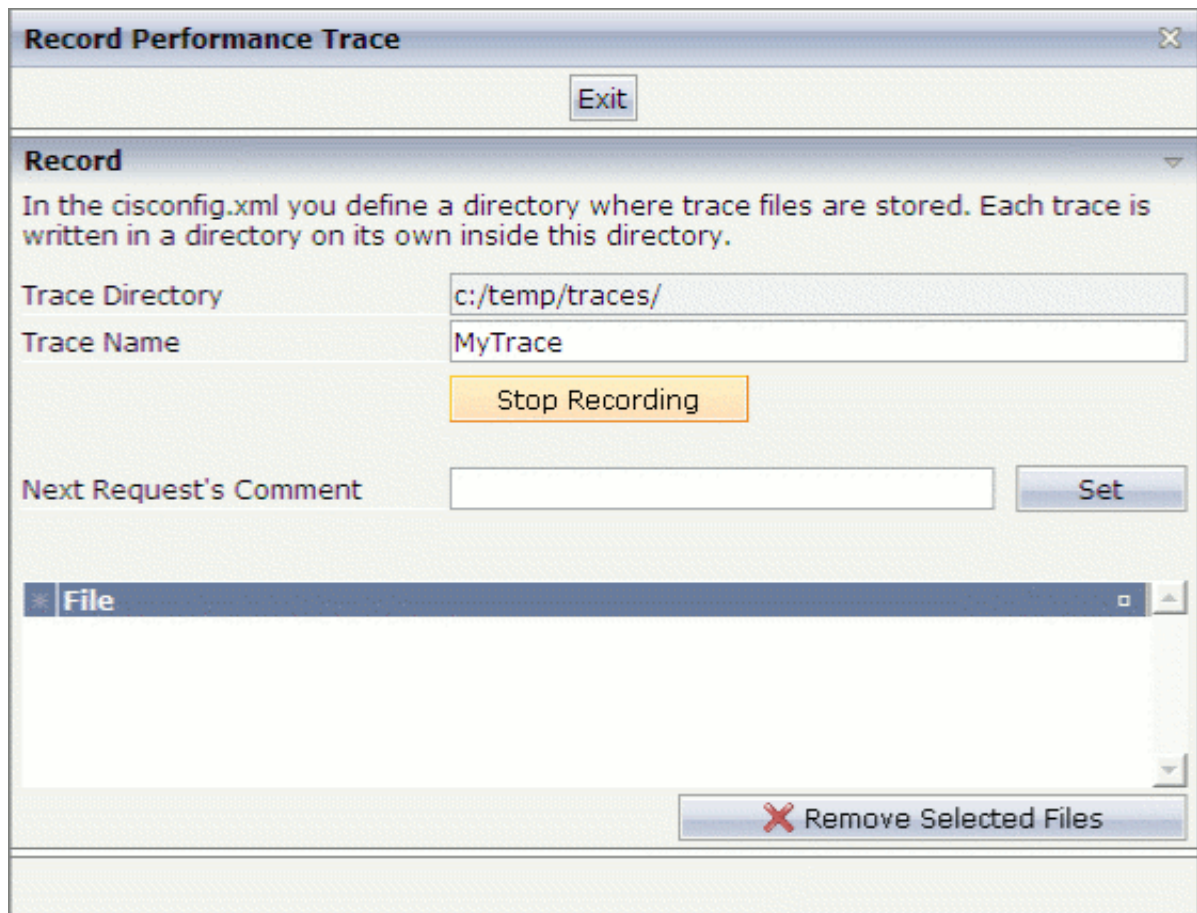
The **Trace Directory** text box indicates the directory in which the traces will be stored. The name of this directory is defined in the file *cisconfig.xml*.

Each trace is written into a separate subdirectory of the trace directory. The name of the subdirectory is determined by your specification in the **Trace Name** text box.

 **Note:** Make sure that the trace directory is available at the specified location. Otherwise, traces will not be written.

- 2 Enter the name of a subdirectory in the **Trace Name** text box.
- 3 Choose the **Start Recording** button.

Additional information is now shown in the dialog.



- 4 In the **Next Request's Comment** text box, enter a comment name for the current trace.

Using comment names, you can subdivide the trace into several sections. The comment name is used in the names of the files that are created for your server requests (see below).

- 5 Choose the **Set** button.

The comment name that you specified is shown below the **Next Request's Comment** text box.

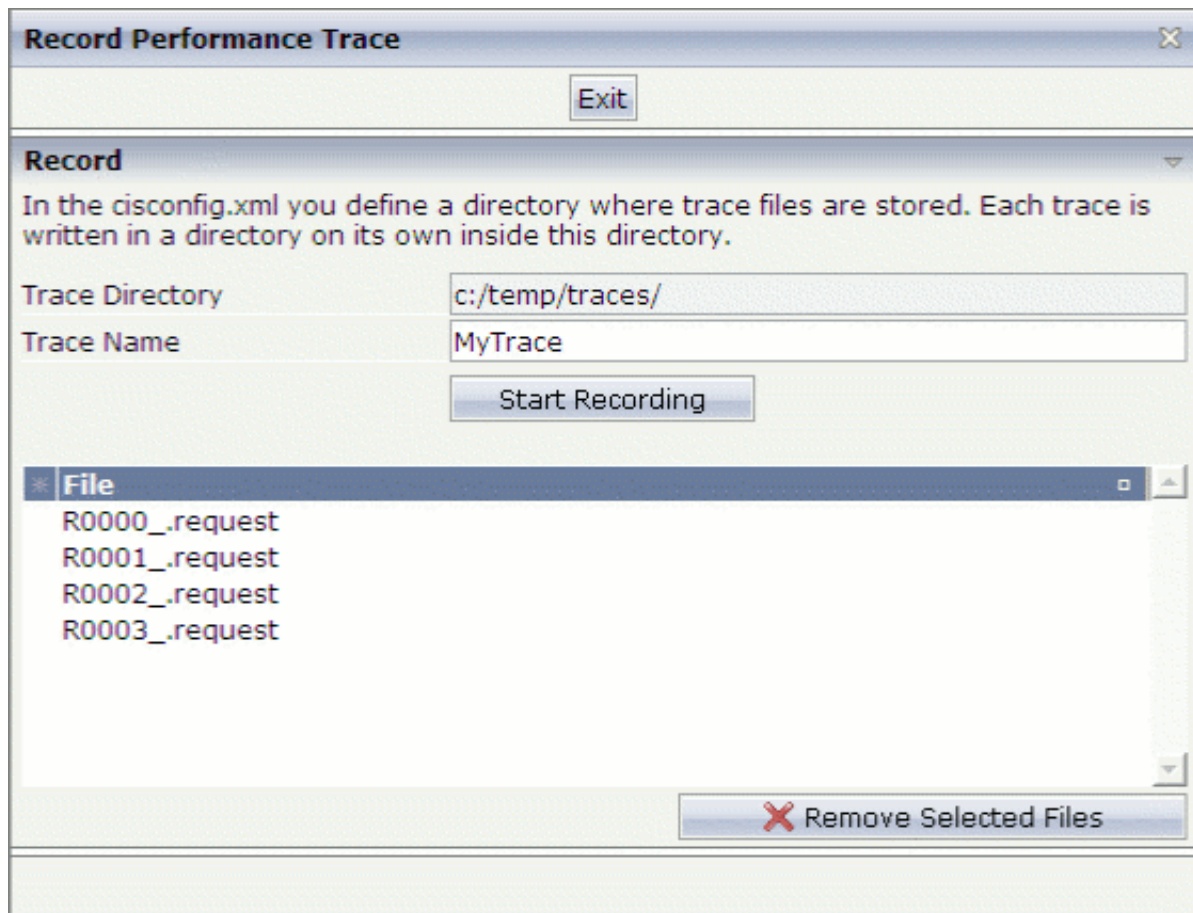
- 6 Initiate an interaction with the server.

For example, open the Layout Manager by choosing the corresponding entry in the navigation frame, and then return to the above dialog by choosing **Start/Stop Trace** in the navigation frame.

All requests (interactions with the server) that you initiate are recorded.

- 7 Choose the **Stop Recording** button.

The files that have been created for your requests are shown in the dialog. The comment name that you specified is part of the file name.



If you want to remove one or more files from the list, select the file(s) and choose the **Remove Selected Files** button.

You can add further requests to this trace by repeating the above steps.

Executing a Performance Trace

Recorded performance traces are executed using the Execute Trace tool.

▶ **To execute a performance trace**

- 1 In the **Performance Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Execute Trace**.

The following dialog appears.

The **Trace Directory** text box indicates the directory in which the tool looks for the recorded traces.

- 2 Specify the following information:

Trace Name

Select the name of the subdirectory which contains the recorded traces for your server requests.

Host to put load on

Enter the URL of the host on which you want to execute the trace.

Repeat trace ... times

Enter the number of times that the trace is to be repeated.

For all defined repetitions, the average process time (in milliseconds) will be shown in the request statistics (see below). The more repetitions you define, the more conclusive is the resulting average value.

Wait ... ms between requests

Enter the waiting time between the requests in milliseconds.

Parallel trace threads

Enter the number of parallel trace threads. This simulates the number of users that access the web application at the same time.

- 3 Choose the **Execute** button.
- 4 Choose the **Refresh View** button to display the request statistics at the bottom of the dialog.

Information on the requests is now shown in the **Request Statistics** section:

Request Statistics

Trace Threads

Working	Finished
0	1

Overview

Request	Calls	OK	ERR	Duration	Min	Max
R0000_.request	10	10	0	7	0	31
R0001_.request	10	10	0	3	0	16
R0002_.request	10	10	0	4	0	16
R0003_.request	10	10	0	4	0	16

Trace file name: c:/temp/traces//MyTrace/traceresults_20070418103259776.csv

The first small table indicates the current number of unfinished and finished trace threads.

The second table provides a statistical evaluation of the executed requests. The following information is provided for each request: the number of calls, the number of successful executions, the number of errors, the average process time for all calls (duration), the minimum

process time for a call, and the maximum process time for a call. The times are given in milliseconds.

In the case of an error, an additional table is shown which provides details on the error.

You can choose the following button in the **Request Statistics** section to display the request statistics in PDF:



The trace result is stored in a CSV file. The path to this file is indicated at the bottom of the dialog.

24 Developer Documents and License Information

- Online Documentation 186
- Java API 186
- Control Reference 186
- License 187

Online Documentation

You can display the Application Designer documentation in the content frame.

▶ To display the Application Designer documentation

- In the **Developer Documents** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Online Documentation**.

The overview page of the Application Designer documentation is shown and you can navigate to the topic that you want to read.

Java API

You can display the Java API documentation in the content frame.

▶ To display the Java API documentation

- In the **Developer Documents** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Java API**.

You can now select the name of a class to display information, for example, on the methods that are available for this class.

Control Reference

You can display the control documentation in the content frame.

▶ To display the control documentation

- 1 In the **Developer Documents** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Control Reference**.

The control documentation is shown.

- 2 Make sure that the option button for the application is selected for which you want to display information.
- 3 Select the name of a control to display information on the properties that are available for this control.

License

You can display legal notices for Application Designer in the content frame.

▶ **To display legal notices**

- In the **License Information** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **License**.

Legal notices are now shown.

25

Special Tools

In the browser, the Unicode representation will be used for your projects. Therefore, your files must be in UTF-8 format.

You can use the Unicode Transfer tool to display a list of all text files of a given project which contain ASCII characters with a code higher than 127.



Caution: This tool is only intended for converting non-UTF-8 files to UTF-8. If you use it with files which are already in UTF-8, these files will be destroyed.



▶ **To check the text files**

- 1 In the **Special Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Unicode Transfer**.
- 2 From the **Project** drop-down list box, select the required project.
- 3 Choose the **UTF-8 Check** button.


When a file is found in the specified project which contains ASCII characters with a code higher than 127, the result is displayed as shown in the following example:

Check	Difference	File
CRIT	ÿ - □	Layout Definitions D:/wrksvam/Development/HTMLBasedGUI_CIS_23/src/webapp
TOTAL		Multi Language Files Number of critical files = 1

The following information is provided in the **Difference** column:

Left Side	Right Side
Non-Unicode representation.	Unicode representation.

- Examine the result list thoroughly. This list does not tell you whether a file is already in UTF-8. It is important that you know the encoding of your files. If a listed file is already in UTF-8, do not try to change its encoding using the **Convert Selected** button, no matter what is shown in the **Difference** column.
- To convert a non-UTF-8 file to UTF-8, select the corresponding row and choose the **Convert Selected** button. This converts the entire file (not only the character that is shown in the **Difference** column).

 **Caution:** Choose the **Convert Selected** button only once per non-UTF-8 file. Otherwise, you will destroy the file.

